Karlsruhe Institute of Technology

KASTEL

Estimating the Cost of Superposition Attacks on Lightweight Cryptography on Fault-Tolerant Quantum Systems

Master's Thesis of

Florian Hägele

at the Department of Informatics
KASTEL – Institute of Information Security and Dependability

Reviewer:	Prof. Dr. Jörn Müller-Quade
Second reviewer:	Prof. Dr. Thorsten Strufe
Advisor:	M.Sc. Marcel Tiepelt
Second advisor:	M.Sc. Astrid Ottenhues

23. June 2021 – 23. December 2021

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 23.12.2021

. . . .

(Florian Hägele)

Abstract

We will propose different quantum superposition attacks on lightweight cryptographic primitives using variants of Simon's algorithm. Three of our attacks are against the NIST Lightweight Cryptography Standardization Process finalist Elephant. The other primitives are LightMAC and ESTATE. We will also show that truncating periodic 2-to-1 functions will not constrain Simon's algorithm. This result can then be used to speed-up existing attacks.

The resource cost of all presented attacks will then be estimated considering a fault-tolerant surface code based quantum computer. We will demonstrate the differences between an adversary who is able to query the Elephant primitive in superposition and one who can only perform classical queries. Even when both have access to the same local quantum computer, the one who has superposition access will recover the secret key in about 10^{11} logical-qubit-cycles and 21.2 seconds, while the other one needs about 10^{20} logical-qubit-cycles and 209.9 years.

Zusammenfassung

Wir werden verschiedene Angriffe in Quantensuperposition auf sogenannte Lightweight-Kryptographie Primitive unter Verwendung von Simon's-Algorithmus vorstellen. Drei unserer Angriffe richten sich gegen den Finalisten des NIST Lightweight Cryptography Standardization Process, Elephant. Die anderen Primitive sind LightMAC und ESTATE. Wir werden auch zeigen, dass das Kürzen der Ausgabe von periodischer 2-zu-1-Funktionen Simon's Algorithmus nicht einschränkt. Dieses Ergebnis kann genutzt werden, um bestehende Angriffe zu beschleunigen.

Die Ressourcenkosten aller vorgestellten Angriffe werden dann unter Berücksichtigung eines fehlertoleranten, auf surface-code basierenden Quantencomputers geschätzt. Wir werden die Unterschiede zwischen einem Angreifer, der in der Lage ist, das Elephant-Primitiv in Superposition anzufragen, und einem, der nur klassische Anfragen stellen kann, demonstrieren. Selbst wenn beide Zugang zum selben lokalen Quantencomputer haben, wird derjenige, der Zugang zu Superpositionen hat, den geheimen Schlüssel in etwa 10^{11} logischen Qubit-Zyklen und 21.2 Sekunden wiederherstellen, während der andere etwa 10^{20} logische Qubit-Zyklen und 209.9 Jahre benötigt.

Contents

List of Figures

List of Tables

Nomenclature

AD	Added Data
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
CAESAR	Competition for Authenticated Encryption: Security, Applicability, and Robustness
CBC	Cipher Block Chaining
CPA	Chosen-Plaintext Attack
CTR	Counter
EM	Even-Mansour
GCM	Galois/Counter Mode
IND	Indistinguishability
IND-CPA	Indistinguishability under a Chosen-Plaintext Attack
IND-qCPA	Indistinguishability under a Quantum Chosen-Plaintext Attack
IoT	Internet Of Things
KIT	Karlsruhe Institute of Technology
LFSR	Linear-Feedback Shift Register
LRW	Liskov, Rivest, and Wagner
MAC	Message Authentication Codes
NIST	National Institute of Standards and Technology
OCB	Offset Codebook Mode
PMAC	Parallelizable Message Authentication Codes
PRF	Pseudorandom Function
PRP	Pseudorandom Permutation

qCPA	Quantum Chosen-Plaintext Attack
QEC	Quantum Error Correction
QECC	Quantum Error Correcting Code
RFID	Radio Frequency Identification
SHA	Secure Hash Algorithm
XOR	Exclusive-Or

1 Introduction

With the help of combining physics, mathematics, and computer science, quantum computing was able to evolve in the last decades from an idealistic concept to one of the most compelling fields of quantum theory. The field of quantum computing is rapidly growing since Paul Benioff [Ben80] proposed the first idea of the quantum mechanical model for the Turing machine in 1980. Since then, there has been a huge increase in the progress of building a large scale quantum computer. Especially in recent years where companies have entered the race to build the first large scale quantum computer [Gib19]. Once large scale quantum computers are built, they will be able to break many of the currently used cryptographic schemes and provides therefore new challenges for the world of cryptography.

The current cryptographic schemes and those that will be standardized must be evaluated for their quantum security. With the rise of Internet of Things and the growing usage of small devices so called Lightweight cryptography is becoming more and more important. This thesis will focus on this field of cryptography and evaluate the quantum security of different Lightweight cryptographic candidates.

1.1 Motivation

In 1994, Peter Shor discovered [Sho94] that efficient quantum algorithms for algebraic problems, like integer factoring and discrete logarithm, exist. This discovery has changed cryptography significantly. Many of the public key cryptographic primitives' security used today rely on the hardness of these problems. Nevertheless, those problems will expire as soon as powerful enough quantum computers are accessible.

These results have shown that there is the need for new standards, especially in the public key cryptography as well as re-evaluations of current standards in the presence of potential quantum attacks. The National Institute of Standards and Technology has organized competitions and challenges to provide new standards for public key encryptions, key exchanges and digital signatures that are quantum secure [ST20b].

Besides the exhaustive search provided by Grover's Algorithm [Gro96], the symmetric cryptography was long time considered to be secure against quantum attacks. Therefore whitening keys or increasing the key size in the used schemes was believed to be enough to provide the needed security. But with the introduction of new attack models and therefore creating more powerful adversaries by giving them the ability to issue quantum queries to an oracle, new cryptoanalyses and attacks were made possible [Dam+11; Kap+16; BJ20; Haa20].

With Simon's period finding Algorithm [Sim97] in 1997 and being able to query superposition states to an encryption oracle, the first exponential speed-up for recovering the boolean hidden shift of a periodic function was found in symmetric cryptography. This led to a variety of new attacks and many cryptoanalyses have shown that precautionary measures against quantum attackers, like whitening keys, are not providing additional security in the quantum CPA setting [LM17].

Another important aspect of quantum security is that known attacks, like Shors's algorithm, can be applied to today's information. An adversary can therefore gather information and apply the attack in a future where powerful enough quantum computers are available. The quantum security of a cryptographic scheme is especially important if it is required to provide long-term security.

It is therefore as important as knowing that an attack is possible to compute its cost and needed effort. This gives the opportunity to estimate the threat and likelihood of the attack and what resources would be necessary to carry it out. The cost of many attacks on symmetric cryptography are only known vaguely, it is thus important to analyze the resource cost of these attacks. This also includes to provide and optimize quantum designs and circuits that implement said attacks.

1.2 Contribution

In this thesis we will analyze the effects of truncating 2-to-1 functions and their impact on Simon's algorithm in section 3.4. We will show, that this impact is lower than assumed. We conclude that truncating periodic functions will not affect the applicability of Simon's algorithm to retrieve the hidden period. This result can be applied to existing superposition attacks to provide exponential speed-up. This is the case for the superposition attack on `LightMAC` presented by Jonas Haas in [Haa20].

In section 5.2 we will propose three new quantum key recovery attacks on the `NIST` Lightweight Cryptography Standardization Process finalist `Elephant` . As well as present the attacks on `LightMAC` and `ESTATE` by Jonas Haas in [Haa20]. All attacks are realized using variants of Simon's algorithm.

In chapter 6 we will estimate the resource cost required to implement these superposition attacks on a fault-tolerant surface code based quantum computer. We consider per-gate error rates from 10^{-3} up to 10^{-5}.

1.3 Related Work

Ivan Dåmgard et al. [Dam+11] defined superposition attacks in 2011 and classified different models, which define how and whether an attacker is able to access a primitive (or an oracle) parametrized with an unknown key. These models are often referred to as Q0, Q1 and Q2 models (e.g. by Xavier Bonnetain in [Bon19, Section 1.3.2]). In their work Dåmgard et al. discuss the security of different classical cryptographical primitives, where the adversary is able to request classical queries in superposition. They analyze the security of zero-knowledge proofs, secret-sharing schemes and multi-party computation.

Kuwakado and Morii presented in [KM10] a quantum three-round-Feistel cipher distinguisher. No classical polynomial algorithm is known up to this date to differentiate

between a three-round-Feistel cipher and a random function. They managed that by cleverly exploiting the Feistel structure to construct a periodic function and using Simon's algorithm afterwards.
In 2017 Santoli and Schaffner extend the idea of Kuwakado and Morii in [SS17, Section 3] by presenting a proof for a more general approach of the internal function of the Feistel cipher. While Kuwakado and Morii assumed a permutation as its internal function, Santoli and Schaffner are capable of managing arbitrary functions.

Having access to an oracle also allows to introduce new security definitions similar to those in the classical world. Boneh and Zahndry gave formal definitions in [BZ13] of indistinguishable under a quantum chosen message attack and other security models. They also concluded that the IND-qCPA is strictly stronger than IND-CPA under the assumption that pseudorandom functions exist. Encryption schemes which are still considered secure in the Q2 model are called quantum secure.

With the new quantum attack models and being able to perform quantum encryption queries Kuwakado and Morii presented an attack on the Even-Mansour construction in [KM12], a construction which is still up to the date of writing this considered secure in the classical world. They demonstrated that by using Simon's Algorithm and having quantum access to an encryption oracle, the Even-Mansour construction collapses.

In 2016 Kaplan et al. [Kap+16] researched applications for Simon's Algorithm and focused on attacks using quantum period finding on symmetric primitives. They show that known classical attacks based on finding collisions can be speed-up using Simon's Algorithm. They manage to completely break different and commonly used block cipher MACs, like CBC-MAC, PMAC, GMAC, GCM, and OCB. In their work they describe forgery attacks on these modes and different CAESAR [ST16] candidates as well. In the superposition attack model Kaplan et al. describe a way to apply Simon's Algorithm on a LRW construction, named after Liskov, Rivest and Wagner, to turn the block cipher into an tweakable block cipher. This allows them to construct a distinguisher between a LRW construction and an ideal tweakable block cipher. At the end of their paper, they apply Simon's Algorithm to slide attacks, managing to achieve an exponential speed-up of the classical cryptanalysis technique.

The use of whitening keys to increase the key length was long time considered a good method to provide additional security against quantum attackers. This was proven false in the quantum CPA setting by Leander and May in their work [LM17]. They combined the quantum algorithms Grover and Simon to show that the needed quantum steps to break the FX-construction is about $O(m + n) \cdot 2^{m/2}$, where m and n are the used key sizes for whitening and internal block cipher. This is about the same amount needed to break the construction without the whitening keys. This showed that using whitening keys does not increase the security against quantum attackers in this setting significantly.

In his master thesis [Haa20] Jonas Haas described superposition attacks against three Lightweight Cryptography candidates Elephant AEAD, LightMAC and ESTATE AEAD. Haas presented a key recovery attack on Elephant AEAD which is at the time of writing a finalist of the `NIST` lightweight cryptography standardization process [ST20a]. Elephant is based on an Even-Mansour construction which is classically considered secure but broken in the quantum world. He then uses a distinction attack to distinguish the LightMAC PRF from a random function which is given as an oracle. This attack applies Simon's

Algorithm and is similar to the one used by Kuwakado et al. on the three-round-Feistel construction presented in [KM10]. The third superposition attack is against the ESTATE AEAD encryption scheme, that is based on the MAC-then-encrypt approach. He targets an underlying tweakable block cipher which is used by ESTATE. The adversary in his attack is not nonce respecting, and therefore able to forge tags and cipher texts. Nonces in superposition are a non trivial topic and discussed by Haas in [Haa20, Section 4]. In this case the nonce is assumed to be constant and the oracle uses the same nonce to answer the queries instead of using random nonces.

In [Bon+19] Bonnetain et al. describe an other attack on the Even-Mansour structure. Remarkable and different to the previous known attacks is that it is done in the Q1 attack model. In their attack, Simon's Algorithm is only applied offline, where $poly(n)$ superposition queries to an oracle are done and stored on n^2 qubits. They are afterwards reused during the iterations of Grover's Algorithm. With the combined use of Grover's and Simon's Algorithm, they were able to break the construction in $\tilde{O}(2^{n/3})$ quantum time, using $\tilde{O}(2^{n/3})$ classical queries and $O(n^2)$ qubits. This was the first time an attack in the Q1 model yielded a quadratic speedup.

As important as finding possible quantum attacks it is to estimate their cost resource cost. This also helps to analyze and evaluate the long-term prospects of quantum designs. The resource cost estimation can be done considering only the algorithm itself or additionally taking quantum error correction into account. It is therefore important to examine how a quantum computer architecture is constructed and where resources are required. The following works have addressed these issues as well as presenting resource cost estimations to given attacks.

Jones et al. have considered what efforts are required to build a large scale quantum computer in [Jon+12]. They introduced a layered quantum computer architecture and differentiate it into five separate layers, where a lower level layer provides services to a higher level one. Beginning from the lowest layer, the five layers differentiate into the Physical, Virtual, Quantum error correction, Logical and Application layer. The Physical layer describes how to store information between processing steps and how to control and measure it. The Virtual layer makes the Physical layer more robust against systematic errors. In the Quantum error correction layer reversible circuits are embedded into error correcting codes by using surface codes. This yields fault tolerant logical qubits and gates. In the Logical layer the fault tolerant gates and qubits are optimized and turned into a logical substrate for universal computing. In the Application layer quantum algorithm are executed with the exertions of the previous layers to supply any arbitrary gates.

In 2020, Xavier Bonnetain and Samuel Jaques presented quantum circuits that implement offline Simon's Algorithm in [BJ20]. This regards a quantum adversary who is limited to only classical queries and perform offline quantum computations. They presented attacks and their cost estimates against the block cipher PRINCE, the MAC Chaskey and the `NIST` lightweight candidate AEAD scheme Elephant as well. They came to the conclusion, that the needed qubits for the given attacks are in the order of 2^{14} qubits. Which is comparable to the amount needed trying to break RSA-2048 where about $2^{12.6}$ qubits are necessary. The used query limits for comparison are provided by Gidney and Ekerå in [GE21]. The attacks on PRINCE and Chaskey are the current most efficient ones, up to the date of writing this. They also state, that since Elephant has a key smaller than its base state, their presented attack is more expensive than an exhaustive search.

In the work of Amy et al. [Amy+17] they study the pre-image cost of the exhaustive quantum key search attack using Grover's Algorithm on SHA-2 and SHA-3 families of hash functions. They measure their cost analysis in surface code cycles. The resulting cost is calculated by the number of logical qubits times depth of the circuit and given in units of surface code cycles. They come to the result that the pre-image attack on SHA2-256 requires about $2^{12.6}$ logical qubits and is around $2^{153.8}$ surface code cycles deep, which results in an overall cost of $2^{166.4}$ logical qubit cycles. For the SHA3-256 attack about 2^{20} logical qubits are needed and is approximately $2^{146.5}$ surface codes cycles deep. This results in a total cost in the order of $2^{166.5}$ logical qubit cycles. Both attacks issue queries to a quantum black-box model and about 2^{128} queries are needed for each of the attack. They conclude that the studied attacks are by far more expensive than a simple query analysis would suggest. Both attacks are about 275 billion times more expensive than taking a query analysis in account.

2 Preliminaries

In this chapter basic and frequently used notations for both the classical and quantum world will be defined. Afterwards important elements for quantum computing will be presented. These includes the basics of quantum mechanics, quantum circuits, and Grover's algorithm [Gro96]. Afterwards, the definition of a layered quantum computer architecture developed by Jones et al. in [Jon+12] will be given. The quantum attack models which are being used to describe the power of an adversary are given at the end of this chapter.

2.1 Basic notations

The description and notations are based on [KL15] by Katz and Lindell as well as [Bon19] by Bonnetain and the lecture notes [The16] by the KIT cryptography and security group. The set of bitstrings of length n will be denoted by $\{0,1\}^n$. The $\oplus$ operator is the bitwise exclusive-or (XOR) of single bits or entire bitstrings, where in the latter case the XOR operation is applied bitwise. Let $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, then $x \,\|\, y \in \{0,1\}^{n+m}$ denotes the concatenation of x and y. If A is a randomized algorithm, $A(x)$ denotes running A with input x and $A(x;r)$ denotes running A with input x and randomness r. The set $\{1,\ldots,n\}$ will be denoted by $[n]$. Choosing an element x uniformly at random from the set R will be noted by $x \overset{\$}{\leftarrow} R$. $\mathcal{A}$ denotes an adversary and $\mathcal{D}$ a distinguisher. $\Pr[X]$ defines the probability of the event X to happen.
A function f is negligible if for all constants $c \in \mathbb{N}$ there exists an integer k such that for every $k_0 > k$ it holds that $|f(k_0)| \leq \frac{1}{k_0^c}$. A function f is called periodic if there exists a s such that for all inputs x, $f(x) = f(x \oplus s)$ holds. $poly$ denotes a polynomial, which means that if $f(n) = poly(n)$ there exists a polynomial p such that $f(n) = O(p(n))$.

2.2 Quantum computing

All vector spaces are assumed to be of finite dimension, otherwise it will be noted. The basic definition and notations of quantum computation are based on the work of Boneh and Zhandry in [BZ13], Nielsen and Chuang in [NC01], and the lecture Notes by Wolf in [Wol19].

2.2.1 Quantum Mechanics

Superposition A quantum system is considered a complex Hilbert space $\mathcal{H}$ together with an inner product $\langle\cdot|\cdot\rangle$. Consider a system that can be in N different, mutually exclusive classical states, represented by the vectors $|0\rangle, \dots |N-1\rangle$. The quantum state of the system is then described by the complex vector

$$|\phi\rangle = \sum_x \alpha_x |x\rangle,$$

of unit norm $\langle\phi|\phi\rangle = 1$. The complex coefficients α_x are called the amplitude of $|x\rangle$ in $|\phi\rangle$ and $|\phi\rangle$ is called the quantum superposition of $|x\rangle$. Thus, the system in quantum state $|\phi\rangle$ is in all classical states at the same time and each is in state $|0\rangle$ with amplitude α_0, in state $|1\rangle$ with amplitude α_1 and so forth. Assuming that $\mathcal{H}$ is a $N-$dimensional Hilbert space, the quantum state $|\phi\rangle$ is then a vector in this space written as

$$|\phi\rangle = \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{N-1} \end{pmatrix},$$

and called a `ket`. The conjugate transpose `bra` is the row vector

$$\langle\phi| = \left(\alpha_0^*, \dots, \alpha_{N-1}^*\right).$$

The inner product between two states $\langle\phi, \psi\rangle$ is equivalent to the dot product between the `bra` and the `ket` vector $\langle\phi|\psi\rangle$.

The joint quantum state of two Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ is defined by their tensor product $\mathcal{H}_1 \otimes \mathcal{H}_2$. Therefore the product state of the vectors $|\phi_1\rangle \in \mathcal{H}_1$ and $|\phi_2\rangle \in \mathcal{H}_2$ is given by $|\phi_1\phi_2\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$.

There are two types of transformation that can be applied to a quantum state. One is the reversible evolution, which is represented by unitary operators of the form $U : \mathcal{H} \longrightarrow \mathcal{H}$, and the other one is measurement, which allows obtaining information about the system.

Measurement in the computational basis Let the quantum state be in superposition $|\phi\rangle = \sum_x \alpha_x |x\rangle$. Measuring this state will return the outcome x with probability $|\alpha_x|^2$, which is the squared norm of the amplitude. Observing a quantum state leads to a probability distribution for the classical states represented by the squared norms of the corresponding amplitudes. Thus,

$$\sum_x |\alpha_x|^2 = 1.$$

Measuring the state $|\phi\rangle$ collapses the superposition of $|\phi\rangle$ to the classical basis state $|x\rangle$. It is not possible to reconstruct the measured quantum state $|\phi\rangle$ from the observed outcome $|x\rangle$. Measurements are therefore non-reversible.

Projective measurement A projective measurement is defined by projectors $P_1, \dots, P_m$, that sum to identity. They are pairwise orthogonal, hence $P_i \cdot P_j = 0$ if $i \neq j$. Let V be a Hilbert space and the quantum state $|\phi\rangle \in V$. The projector P_j projects on some subspace V_j of V. The state ϕ can then be decomposed as

$$|\phi\rangle = \sum_{j=1}^{m} \left|\phi_j\right\rangle, \text{ where } \left|\phi_j\right\rangle = P_j |\phi\rangle \in V_j.$$

Since the projectors $P_1, \ldots, P_m$ are pairwise orthogonal, both the subspaces V_j as well as the quantum states $|\phi_j\rangle$ are orthogonal. Applying the measurement to the quantum state $|\phi\rangle$ yields the outcome j with probability $\| \, |\phi_j\rangle \, \|^2 = \langle\phi| P_j |\phi\rangle$. Given that the outcome j occurred, the measured state will immediately transition (collapse) into the new state

$$\frac{P_j |\phi\rangle}{\|P_j |\phi\rangle \|}.$$

Any orthonormal basis $B = |\psi_0\rangle \ldots, |\psi_{N-1}\rangle$ can now be considered as the projective measurement defined by the projectors $P_j = |\psi_j\rangle \langle\psi_j|$. That process is called measuring in basis B. If the quantum state $|\phi\rangle$ equals to one of the basis vectors $|\psi_j\rangle$, the measurement outcome will then yield j with probability 1.

Observables A projective measurement with projectors $P_1, \ldots, P_m$ and associated distinct outcome $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$ can be written as the matrix $M = \sum_{i=1}^m \lambda_i P_i$.
M is called an observable and is a Hermitian, $M = M^*$, operator on the state space of the system being observed. The observable has therefore a spectral decomposition and can be written as above with P_i being the projectors into the eigenspace of M with eigenvalues λ_i.

This has the advantage, that it is easy to calculate average values of the outcome. By measuring the state $|\phi\rangle$, the probability of the outcome λ_i is $\|P_i |\phi\rangle = \langle\phi| P_i |\phi\rangle$. The average value, or estimated values, of the measurement is hence

$$\begin{aligned}\mathbb{E}(M) &= \sum_{i=1}^m \lambda_i \langle\phi| P_i |\phi\rangle \\ &= \langle\phi| \left(\sum_{i=1}^m \lambda_i P_i\right) |\phi\rangle \\ &= \langle\phi| M |\phi\rangle .\end{aligned}$$

Unitary evolution Instead of measuring the quantum state $|\phi\rangle$ and therefore destroying its superposition, it is possible to apply some linear operation and converting it into another quantum state

$$|\psi\rangle = \sum_{x=0}^{N-1} \beta_x |x\rangle .$$

To apply a linear operation that converts the state $|\phi\rangle$ to $|\psi\rangle$, is equivalent to multiplying $|\phi\rangle$ with a complex valued $N \times N$ unitary matrix U

$$\begin{pmatrix} \beta_0 \\ \vdots \\ \beta_{N-1} \end{pmatrix} = U \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{N-1}, \end{pmatrix}$$

or $|\psi\rangle = U |\phi\rangle$. An unitary operator over a N-dimensional Hilbert space $\mathcal{H}$ is a $N \times N$ matrix U such that $UU^\dagger = I_N$, where $U^\dagger$ is the complex conjugate transpose of U. It preserves the norm of the vectors, hence $\sum_x |\beta_x|^2 = 1$. Since an unitary transformation always has an inverse, it follows that any (non-measuring) operation on quantum states has to be reversible. The effect of U can always be reversed by applying U^{-1} to the result.

An important theorem in quantum mechanics considering whether a state can be copied is the quantum no-cloning theorem.

Theorem 1 (Wootters in [WZ82]). *There does not exist an unitary evolution U, given an arbitrary unknown state $|\phi\rangle |0\rangle$, that maps*

$$U(|\phi\rangle |0\rangle) = |\phi\rangle |\phi\rangle .$$

2.2.2 Qubits and quantum registers

The qubit In classical computation bits are used as unit of information, they can either take one of the states 0 or 1. In quantum computation the unit of information is the qubit (quantum bit) which can be described as an element of the two-dimensional complex Hilbert space $\mathcal{H} = \mathbb{C} \times \mathbb{C}$. It is a superposition of the two computational basis states $|0\rangle$ and $|1\rangle$.

$$|\varphi_1\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle ,$$

where α_0 and α_1 are defined as in subsection 2.2.1.

Multiple qubits Suppose having two bits in the classical world. The number of possible states would be four $00, 01, 10$ and 11. Similarly, the 2-qubit system can be considered to have the 4 computational states $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$. A pair of qubits exists therefore in a superposition of these basis states described by the quantum state vector

$$|\varphi_2\rangle = \alpha_{00} |00\rangle + \alpha_{01} |01\rangle + \alpha_{10} |10\rangle + \alpha_{11} |11\rangle .$$

Like in the previous section it holds that the sate $|x\rangle$ occurs after measuring $|\varphi_2\rangle$ with probability $|\alpha_x|^2$ and $\sum_{x\in\{0,1\}^2} |\alpha_x|^2 = 1$. In general, a quantum register of n qubits has 2^n many different base states $|0\rangle, \ldots, |2^n - 1\rangle$ and can be in a superposition

$$\sum_{x=0}^{2^n-1} \alpha_x |x\rangle . \tag{2.1}$$

Measuring this state yields the $n-$bit state $|x\rangle$ with probability $|\alpha_x|^2$. Measuring only a subset of the qubits would correspond to a projective measurement. Assuming only the first qubit is measured, the two projectors P_0 and P_1 are then defined as

$$P_0 = |0\rangle \langle 0| \otimes I_{2^{n-1}} \text{ and } P_1 = |1\rangle \langle 1| \otimes I_{2^{n-1}}.$$

Therefore, measuring m bits of the n-qubit state would correspond to the projective measurement having 2^m projectors $P_i = |i\rangle \langle i| \otimes I_{2^{n-m}}$, for $i \in \{0, 1\}^m$.

Entanglement and the Bell state An important 2-qubit register is the so called Bell state or EPR-pair[1] that is in the state

$$\frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) .$$

[1]In honor of Einstein, Podolsky, and Rosen

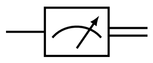
The special property that the Bell state has is that upon measuring the first qubit and observing a $|0\rangle$, the whole state collapses to $|00\rangle$ and when observing a $|1\rangle$ the state collapses to $|11\rangle$. Hence, measuring the second qubit gives always the same result as the first measurement qubit. In other words, measuring the first qubit immediately fixes the outcome of the second unmeasured qubit to a classical value, the measurement outcome is therefore correlated.

Let $\mathcal{H}_A$ and $\mathcal{H}_B$ be two Hilbert spaces, $|0\rangle, \dots, |N-1\rangle$ be the orthonormal basis of $\mathcal{H}_a$ and $|0\rangle, \dots, |M-1\rangle$ the orthonormal basis of $\mathcal{H}_B$. The tensor product space $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$ is a $NM-$dimensional space spanned by the set of states $\{|i\rangle \otimes |j\rangle \,|\, i \in \{0, \dots, N-1\}, j \in \{0, \dots, M-1\}\}$. A state $|\phi\rangle$ in $\mathcal{H}$ is called bipartite. $|\phi\rangle$ is referred to as entangled if it cannot be written as a tensor product $|\phi_A\rangle \otimes |\phi_B\rangle$, where $|\phi_A\rangle$ is a state in $\mathcal{H}_A$ and $|\phi_B\rangle$ in $\mathcal{H}_B$.

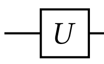
2.2.3 Quantum circuits and gates

In the classical world computers are built from electrical circuits containing logic and wires. A quantum computer is composed of quantum circuits, which contain wires and elementary gates that carry and manipulate the quantum information.

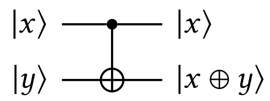
An example of different quantum circuits can be found in Figure 2.1. The quantum wires will be represented by a single line, classical wires by a double line and a single qubit unitary gate is denoted by a rectangle with the gates name in it. This can be seen in Figure 2.1a and Figure 2.1b. If the input or output registers of the quantum circuit need to be specified, it will be noted at the corresponding positions of the quantum circuit, this is denoted in Figure 2.1c. If the quantum register is of size n, it is noted on the input register, see Figure 2.1d.



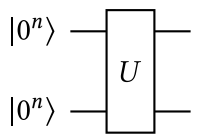
(a) Quantum measurement gate



(b) Single qubit unitary gate



(c) CNOT gate



(d) Quantum circuit operating on 2 $n-$qubit registers

Figure 2.1: Examples of quantum circuits.

Single qubit gates An unitary linear operator that manipulates a small amount of qubits is called a quantum gate. The Pauli gates I, X, Y and Z are all Single qubit gates. They are represented by the 2×2 unitary matrices $\sigma_x, \sigma_y, \sigma_z$

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X = \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \text{ and } Z = \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The X-gate is often also referred to as the NOT- or bit-flip-gate since it swaps $|0\rangle$ and $|1\rangle$ of the qubit. The phaseflip-gate Z leaves $|0\rangle$ unchanged and flips the sign of $|1\rangle$. The Pauli matrices also satisfy $XZ = iY$ and the square of a Pauli matrix is the identity matrix I. The Pauli matrices are Hermitian, examples for 2-dimensional observables with eigenvalues ± 1, and mutually anticommute meaning $[A.B] = AB - BA \neq 0$ for Pauli matrices A and B. The Pauli group on a Single-qubit is defined as the multiplicative subgroup generated by $\mathcal{P} = \{iI, X, Y, Z\}$ [NC01] and can be extended to the Pauli group on n qubits $\mathcal{P}_n$.
One important observation is that any 2×2 matrix M can be expressed using a linear combination of the Pauli-matrices

$$M = \alpha_0 I + \alpha_1 X + \alpha_2 XZ + \alpha_3 Z. \tag{2.2}$$

One of the most important single qubit gate is the Hadamard transformation H. Represented by the unitary matrix

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Applying H to a basic state $|0\rangle$ turns it into a superposition, which when measured, collapses with equal probability to $|0\rangle$ or $|1\rangle$. The same applies for $|1\rangle$. The $H|0\rangle$ state is also defined as the $|+\rangle$ state and $|-\rangle$ for $H|1\rangle$.

$$H|0\rangle = |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \text{ and } H|1\rangle = |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Applying the Hadamard gate a to $|+\rangle$ or $|-\rangle$ transforms it back to $|0\rangle$ and $|1\rangle$. This shows that $H^2 = I$. Applying the $n-$folded Hadamard gate $H^{\otimes n}$ to a register of n qubits in the zero state, converts the register into

$$H^{\otimes n}|0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle, \tag{2.3}$$

that is a superposition of all 2^n $n-$bit strings. Applying the $n-$folded Hadamard gate to an arbitrary state $|y\rangle$, with $y \in \{0,1\}^n$ results in

$$H^{\otimes n}|y\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} |x\rangle, \tag{2.4}$$

where $x \cdot y$ denotes the inner product of the $n-$bit strings x and y.

Another set of single qubit gates are the phase shift gates. They map the basic states $|0\rangle$ to $|0\rangle$ and rotate the $|1\rangle$ state to $e^{i\theta}|1\rangle$ by an angle of θ. The Pauli Z-gate can be considered as a rotation by $\theta = \pi$. Important member of this set are the T and phase gate S.

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \text{ and } S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

The $Z-$, $T-$ and $S-$gates satisfy $T^2 = S, S^2 = Z$ and $S^4 = I$.

Multiple qubit gates The controlled-NOT gate, or CNOT gate, is a 2-qubit gate that negates the second bit of input if the first qubit is 1, otherwise it leaves the second bit unchanged. The first qubit can therefore be seen as the control bit and the second as the target qubit. The quantum CNOT circuit can be found in Figure 2.1c

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

The action of this gate can be summarized as $|x\rangle|y\rangle \longrightarrow |x\rangle|x \oplus y\rangle$. The CNOT gate can be seen as the reversible generalized-XOR gate.

The Clifford group is the set of unitary operators U such that $UPU^\dagger \in \mathcal{P}_n$, where P_n denotes the Pauli group. U is called the normalizer of $\mathcal{P}_n$. The Clifford gates are then defined as the elements of the Clifford group. The set of the Hadamard gate H, phase gate S and CNOT gate generate the Clifford group [Got98b].

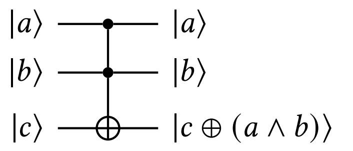
An important theorem that will later be used for Stabilizers and quantum error correcting codes is the Gottesman-Knill theorem.

Theorem 2 (Theorem 1 in [Got98a]). *A quantum circuit performing only the following elements*

1. *Clifford gates,*
2. *Preparation of qubits in computational bases,*
3. *Measurements in the computational bases*

can be perfectly simulated in polynomial time on a probabilistic classical computer.

The Toffoli gate has three input and three output qubits. Similar to the CNOT gate, two of the inputs are control bits and the third qubit is the target. It is flipped if both of the control qubits are 1 and otherwise left unchanged. This can be summarized as $|x, y, z\rangle \longrightarrow |x, y, z \oplus (x \wedge y)\rangle$. It is also called the CCNOT gate. The Toffoli quantum circuit and its unitary matrix representation can be found in Figure 2.2.



$$\text{Toffoli} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Figure 2.2: The quantum Toffoli gate and unitary matrix representation

Universal quantum gate set A finite set of gates that can approximate any unitary operation arbitrarily well is said to be universal for quantum computation or a universal quantum gate set. Although there are many different combinations of gates that meet this criteria, this work will consider only the following variations and refer to the others in [NC01; Wol19].

The set of all Single qubit gates together with the CNOT gate is universal. However, this is not particularly efficient and the number of gates is tried to be kept as small as possible. The Clifford gates are not universal since some gates, like the T-gate, can not be approximated. Gottesman states in [Got98a] that the Clifford gates together with the quantum Toffoli gate or the T-gate form a universal set of gates. The Solovay-Kitaev theorem implies [NC01, Appendix 3] that the set consisting of CNOT-, Hadamard-, phase gate S and the T-gate is sufficient to implement an arbitrary quantum algorithm. This set of gates is often referred to as the Clifford+T gates.

Quantum parallelism One important quantum-mechanical effect is the quantum parallelism. Let $f : \{0,1\}^n \to \{0,1\}^m$ be an arbitrary classical function. The quantum circuit that maps the quantum state $|x\rangle |0^m\rangle$ to $|x\rangle |f(x)\rangle$ for every $x \in \{0,1\}^n$ is denoted by U_f. The first register is called the data-register and the second the target-register. By using Equation 2.4 on the first $n-$qubit register, a superposition of all possible inputs x is obtained. Applying U_f to the state result in

$$U_f\left(\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle |0^m\rangle\right) = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle |f(x)\rangle . \tag{2.5}$$

Thus, the data register is in a superposition of all $f(x)$. This does not seem to be particularly useful at first, since only a random value x and $f(x)$ will be observed when the state is measured. But being able to manipulate the state in specific ways and gain information while the state is still in superposition is very valuable.

2.2.4 Quantum algorithms

Two efficient quantum algorithm, Grover's and Simon's algorithm, had a big impact on symmetric cryptography. Simon's algorithm will be discussed in all its variants in chapter 3. In 1996 Lov Grover presented an efficient quantum search algorithm for databases in [Gro96] that could also be used as an exhaustive search. It solves the following problem:

Grover's problem: Given a test function $f : X \to \{0,1\}^n$, such that the superposition $\sum_{x\in X} x$ is computable in $O(1)$ time. Given oracle access to f and the promise that an unique $x \in X$ exists such that $f(x) = 1$, find x.

In classical computation this problem cannot be solved in fewer than $O(|X|)$ evaluations, Grover's algorithm solves it in $O(\sqrt{|X|})$ evaluations. Therefore Grover's algorithm allows a quadratic speedup on classical exhaustive search.

2.3 Architecture of a quantum computer

Layered Architecture for quantum computing One difference between quantum and classical computation is that quantum computers are only able to implement reversible circuits (logic). While this restriction does not exist in classical computing it is crucial in quantum computing. Thus classical circuits can not be easily translated into quantum circuits. Therefore additional information in the form of extra qubits is oftentimes needed. These extra qubits are called ancilla (qubits). Especially for quantum error correction and creating fault tolerant gates ancillas are very important.

Jones et al. address this issue in [Jon+12] while also considering what efforts are required to build a large scale quantum computer. They introduced a layered quantum computer architecture and differentiate it into five separate layers, where a lower level layer provides services to a higher level one.

Layer 1: Physical The Physical layer describes how to store information between processing steps and how to control and measure it. These units used in this layer are called physical qubits.

Layer 2: Virtual The Virtual layer makes the Physical layer more robust against systematic errors by enforcing symmetries in the system to cancel out correlation errors. The physical qubits from the previous layer are formed into a two-state system, the virtual qubit, resembling an ideal qubit.

Layer 3: Quantum Error Correction For quantum computing it is important to have fault-tolerant quantum error correction (QEC) that can handle errors . While the Virtual layer let correlation errors cancel themselves out, the QEC layer however isolates and eliminates arbitrary errors as long as the error rate is below a certain threshold [AB08]. The errors that are prevented in this layer are caused by quantum decoherence and quantum noise. In the classical world redundancy is used to address this issue, but since copying an arbitrary quantum state is proven to not be possible by the no-cloning theorem [WZ82] and [NC01, Section 1.3.5], the quantum information must be preserved alternatively. There are many ways to address this issue and Jones et al. have chosen to analyze the case of stabilizer code, in particular the surface code for its high threshold [Fow+12] and two-dimensional nearest-neighbour geometry.

An important aspect in the QEC layer is the threshold error rate of error-correcting code. It is defined as the breaking point of the error rate at which the error correction starts to grant a net gain in protecting information. A quantum system which operates below the threshold is called a functioning quantum system and a practical quantum system if it operates well below the threshold.

Another aspect of quantum error correction is how errors are tracked by Pauli frames [Kni05; DA07] in classical hardware. A Pauli frame mechanism is a classical technique to track the result of applying a set of Pauli gates to single qubits. QEC codes project the encoded state into a perturbed code word with faulty single qubit Pauli gates applied. The syndrome reveals what these Pauli errors are and error correction is achieved by applying the particular same Pauli gates to the appropriate qubits.

The surface code constructed by virtual qubits and gates in this layer yields fault-tolerant logical qubits and gates for the Logical layer.

Layer 4: Logical The fault tolerant gates and qubits coming from the QEC layer are taken and made into a logical substrate for universal computing. Therefore additional processing of error corrected gates and qubits is required to produce arbitrary gates that are necessary for the Application layer. The full Clifford group, a series of transformations that effect permutations of the Pauli gates, is not provided in the surface code without some sort of ancilla. One of the missing gates of the Clifford group is the $T-$gate [FSG09; NC01].

The set of fault-tolerant gates which are generated without the use of any ancilla by the QEC Layer are called *fundamental gates*. The Logical layer is able to construct arbitrary gates by taking circuits of fundamental gates and injecting ancillas in error correcting code. The surface codes architecture is for example able to create the missing phase gate S of the Clifford group [FSG09; BK05a; RHG07]. This is done by injecting and purifying specific ancilla states and then consuming these ancillas to produce the gate without measurement.

Another important function of the Logical layer is the distilling into high fidelity ancilla states. The distilled states are then used to construct arbitrary gates with specific quantum circuits [FSG09; BK05a; RHG07].

Layer 5: Application In the Application layer quantum algorithm are executed. The exertions of the previous layers to supply any arbitrary gate allow the Application layer to not be worried about the implementation details of the quantum computer. It is therefore an optimal programming environment. The Application layer is composed of application gates and qubits, which are logical qubits explicitly used by a quantum algorithm. The Application layer gates are nearly equivalent to logical gates, they differentiate between what resources are visible to the algorithm and what is hidden in the mechanism of the Logical layer.

Since it is important to analyze and evaluate the long-term prospects of a quantum design, accurate resource estimation for a quantum application are of great interest. When an analysis of a quantum algorithm quotes an amount of qubits without referring to fault tolerant error correction, then this oftentimes means the amount of application qubits needed [Bea02; Zal06].

Even though a quantum algorithm could demand any arbitrary gate, not all have the same resource cost. When ancilla preparation is taken into account, T gates are responsible for more than 90% of the circuit complexity in a fault-tolerant algorithm [Isa+08]. Since the number of Toffoli[2] gates is only dependent on the used algorithm and is not machine-dependent, the resource estimation for an application are counted in terms of Toffoli gates.

[2]The T- and Toffoli-gates are two distinct gates

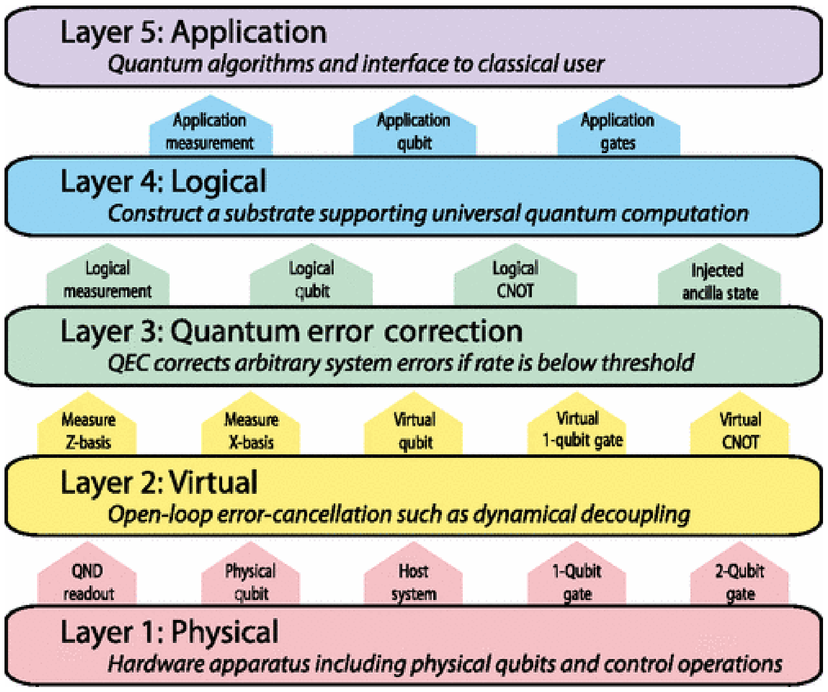

Layer 5: Application
Quantum algorithms and interface to classical user
Application measurement
Application qubit
Application gates
Layer 4: Logical
Construct a substrate supporting universal quantum computation
Logical measurement
Logical qubit
Logical CNOT
Injected ancilla state
Layer 3: Quantum error correction
QEC corrects arbitrary system errors if rate is below threshold
Measure Z-basis
Measure X-basis
Virtual qubit
Virtual 1-qubit gate
Virtual CNOT
Layer 2: Virtual
Open-loop error-cancellation such as dynamical decoupling
QND readout
Physical qubit
Host system
1-Qubit gate
2-Qubit gate
Layer 1: Physical
Hardware apparatus including physical qubits and control operations

Figure 2.3: The layered architecture of a quantum computer, from Jones et al. in [Jon+12]. Layered control stack which forms the framework of a quantum computer architecture. Vertical arrows indicate services provided to a higher layer.

2.4 Quantum attack models

The following classification into different quantum attack models is done following the work of Bonnetain in [Bon19] and Gagliardoni in [Gag17]. The classical world is described by the Q0 model, while the Q1 model defines an adversary having access to a local quantum computer but only classical access to the oracle. The Q2 model describes a model in which the adversary has access to a quantum oracle as well as a local quantum computer.

Classical Computations (Q0): There is no quantum computation at all. The attacker performs only classical computations.

Classical queries (Q1): Attackers have access to a local quantum device to perform offline computations. However, the queries that the attacker is able to perform are classical. This model is considering a post-quantum world where quantum computers are accessible. The goal is to examine constructions that rely on hardness assumption which withhold in the classical world but do not hold against quantum computers. It is believed to be a more realistic model to classical ones since today's information can be collected and retained until a quantum computer is accessible to break them.

Quantum queries (Q2): In this model the schemes are classical and the attackers are quantum like in Q1. In addition the adversaries are always given quantum access to classical oracles. So an adversary can query the oracle in arbitrary superposition of the inputs and receives the superposition of outputs. The oracle can be described as $O_k : \{0,1\}^n \rightarrow \{0,1\}^n$ and the adversary can make quantum queries $|x\rangle\,|y\rangle \mapsto |x\rangle\,|y \oplus O_k(x)\rangle$, where $|x\rangle$ and $|y\rangle$ are the quantum states to their correspondent arbitrary n-bit strings x and y. This is considered to be a much stronger model than the Q1 one. Constructions which are secure in the Q2 model will be called quantum secure.

3 Simon's algorithms

Simon's algorithm was first introduced by Daniel Simon in 1997 in [Sim97]. It is the first example of a quantum algorithm to show an exponential speed-up versus the best known classical algorithm. For a long time, there was no specific application for Simon's algorithm. This changed, as Kuwakado and Morii presented in [KM10] a polynomial distinguisher on the 3-round Feistel construction, that is classically provably secure. This opened the way for other attacks and new variations of Simon's algorithm. In this chapter these variations, that will be used for the superposition attacks presented in chapter 5, will be introduced in the following. Beginning from the basic version of Simon's algorithm to the Grover-meets-Simon algorithm by Leander and May in [LM17]. Afterwards the first variant of Simon's algorithm applicable in the Q1 model called the offline Simon's algorithm by Bonnetain in [Bon+19] will be presented. After that, the impact and applicability of Simon's algorithm on truncated functions is analyzed, this is a result of this thesis.

3.1 Simon's algorithm

Daniel Simon presented in [Sim97] in 1997 his algorithm as a way to address the hidden subgroup problem. It grants a way to find hidden periods in 2-to-1 functions. It solves the following problem:

Simon's problem: Given a Boolean function $f : \{0,1\}^n \to \{0,1\}^n$ and the promise that a $s \in \{0,1\}^n$ exists such that for any $x, x' \in \{0,1\}^n$, with $x \neq x' : [f(x) = f(x')] \Leftrightarrow [x \oplus x' \in \{0^n, s\}]$, find s.

In the classical world this problem is as hard as collision search which can be done in $\Theta(2^{n/2})$ time. In a quantum setting this problem can be solved in linear time and therefore yields an exponential speedup in comparison to the classical world.

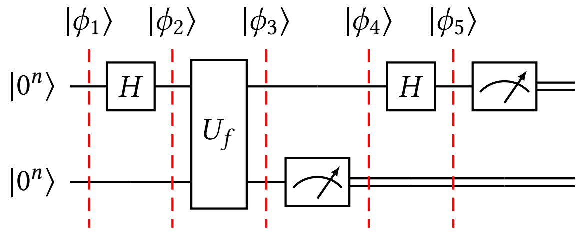


Figure 3.1: Full circuit of Simon's algorithm

The quantum algorithm for Simon's algorithm operates on a $2n$ quantum registers. Two different cases must be considered, the Boolean function f is either periodic in $s \in \{0,1\}^n$

or not. For the sake of convenience, if the function is not periodic, it is assumed to be 1-to-1 and if it is periodic 2-to-1. Non-periodic collision will not be considered. In the case that f is periodic, Simon's algorithm finds enough linear independent vectors to reconstruct s. The procedure is illustrated in Figure 3.1 and repeats the following steps.
In the beginning both registers are in the zero state

$$|\phi_1\rangle = |0^n\rangle \otimes |0^n\rangle . \tag{3.1}$$

Afterwards the Hadamard transformation $H^{\otimes n}$ is applied to the first register transforming it into the quantum superposition

$$|\phi_2\rangle = (H^{\otimes n} \otimes I^{\otimes n}) |\phi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |0^n\rangle . \tag{3.2}$$

Next, one apply the unitary operator U_f corresponding to the function f resulting in the state

$$|\phi_3\rangle = U_f |\phi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle . \tag{3.3}$$

Measuring the second register yields the result $z = f(x)$. If f is not periodic then each of the $f(x)$ are distinct and hence the state in the first register will be a random $|x\rangle$ with equal probability over $x \in \{0,1\}^n$. If on the other hand f is periodic in s then $f(x) = f(x \oplus s) = z$. Hence, the first register is now in a random quantum state but unlike the previous case in $\frac{1}{\sqrt{2}}(|x\rangle + |x \oplus s\rangle)$, $x \in \{0,1\}^n$. The state of the first register can therefore be described as follows

$$|\phi_4\rangle = \begin{cases} \frac{1}{\sqrt{2}}(|x\rangle + |x \oplus s\rangle), & \text{if } s \neq 0^n, \\ |x\rangle , & \text{if } s = 0^n. \end{cases} \tag{3.4}$$

Applying the Hadamard transformation $H^{\otimes n}$ to a state $|b\rangle$ different from $|0^n\rangle$ gives

$$H^{\otimes n} |b\rangle = \frac{1}{\sqrt{2^n}} \sum_{c \in \{0,1\}^n} (-1)^{b \cdot c} |c\rangle ,$$

where $b \cdot c$ defines the inner product modulo 2. If f is a 1-to-1 function the first register is in the state $|x\rangle$. After performing the Hadamard transformation on this state and measuring it afterwards one obtains a random outcome $|y\rangle$ with equal probability. Considering now that f is periodic, applying the second Hadamard transformation to the first register gives the state

$$|\phi_5\rangle = H^{\otimes n} |\phi_4\rangle = \sum_{y \in \{0,1\}^n} \alpha_{z,y} |y\rangle . \tag{3.5}$$

The amplitude $\alpha_{z,y}$ is $\pm\frac{1}{\sqrt{2^n}}$ if f is not periodic and otherwise

$$\begin{aligned} \alpha_{z,y} &= \frac{1}{\sqrt{2^{n+1}}}(-1)^{x \cdot y} + (-1)^{(x \oplus s) \cdot y} \\ &= \frac{1}{\sqrt{2^{n+1}}}(-1)^{x \cdot y}(1 + (-1)^{s \cdot y}). \end{aligned} \tag{3.6}$$

The amplitude of $|y\rangle$ is non-zero iff $s \cdot y = 0 \pmod 2$ and hence y is orthogonal to s. Furthermore, each of the amplitudes are of equal magnitude and thus measuring in the computational basis one will obtain uniformly at random a y such that $s \cdot y = 0$. Non orthogonal y will never be observed. This can be easily verified, since the probability to observe a specific y is equal to the square of its amplitude.

$$\begin{aligned}
\mathbb{P}[y \text{ is observed}] &= \|\alpha_{z,y}\|^2 \\
&= \left\| \frac{1}{\sqrt{2^{n+1}}} (-1)^{x \cdot y} (1 + (-1)^{s \cdot y}) \right\|^2 \\
&= \left\| \frac{1}{\sqrt{2^{n+1}}} (1 + (-1)^{s \cdot y}) \right\|^2 \\
&= \begin{cases} \frac{1}{2^{n-1}}, & \text{if } s \cdot y = 0 \pmod 2 \\ 0, & \text{if } s \cdot y = 1 \pmod 2 \end{cases} .
\end{aligned} \tag{3.7}$$

Repeating these steps $O(n)$ times until $n - 1$ distinct linear independent $y_1, \dots, y_{n-1}$ with $y \cdot s = 0$ are obtained. They span the $n-1$ dimensional space perpendicular to s. Thus, using Gaussian elimination the period s can be uniquely determined. To check for the correct value of s, one compares $f(0) \overset{?}{=} f(s)$ and if the equality holds, the correct value of s was found.

Bonnetain provided tight bounds for all variants of Simon's algorithm in [Bon20]. These results and heuristics will be used throughout this thesis and referred to in the corresponding sections. The proofs and analyses can all be found in [Bon20].

Heuristic 1. *(Heuristic 2 in [Bon20]) Simon's algorithm succeeds in* $n + 3$ *queries on average and with* $n + \alpha + 1$ *queries, it succeeds with probability* $1 - 2^{-\alpha}$.

3.2 Grover-meets-Simon algorithm

Besides the main application of Simon's algorithm to determine the period of a periodic function, there are additional practices. As described above, Simon's algorithm subroutine returns a random vector y if the function f is not periodic. Therefore the linear system of equations will likely have full rank and no solution besides the trivial one. However, if the function is periodic, the linear system of equations has rank $n - 1$. This concept can then be used in quantum distinguishers to check whether a function is periodic or not.

Another application is to implement the basic version of Simon's algorithm reversible by not measuring both registers. Hence, they will both stay in superposition. The value of the first register $|y\rangle$ is still a superposition of values orthogonal to s. Having a circuit to compute linear algebra, one is able to create a reversible implementation of Simon's algorithm. Applying it several times in parallel and using the quantum linear algebra circuit, one is able to determine the correct value of s. The quantum circuit for the linear algebra is presented by Bonnetain and Jaques in [BJ20, Section 5].

The above stated applications combined lead to a new version of Simon's algorithm, named Grover-meets-Simon which was presented by Leander and May presented in [LM17] for quantumly attacking the FX-construction. It addresses the following problem

Finding periodic functions: Given a Boolean function $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ with the promise that there exists a unique $\kappa_0 \in \{0,1\}^k$ such that $f(\kappa_0, \cdot)$ is periodic, find κ_0 and the period s of $f(\kappa_0, \cdot)$.

By using a Grover search over $\{0,1\}^k$, one can identify the correct guess by testing if $f(\kappa_0, \cdot)$ is periodic by using the reversible version of Simon's algorithm. Grover is used as an outer loop. It works by a process called amplitude amplification. Each iteration the amplitude of success is increased by a constant, the probabilities are the square of their amplitudes. Thus, one has to repeat the process approximately $O(2^{k/2})$ times. Simon's algorithm functions as an inner loop and needs about $O(n)$ queries and $O(n^3)$ running time for performing the linear algebra.

For the amplifying part of the Grover-meets-Simon algorithm on a quantum algorithm $\mathcal{A}$ operating on q qubits, a function $\mathcal{B} : \{0,1\}^q \to \{0,1\}$ that classifies the outcome of $\mathcal{A}$ as good or bad is needed, see [Bra+02, Theorem 2]. Define the unitary operator $Q = -\mathcal{A}S_0\mathcal{A}^{-1}S_{\mathcal{B}}$, where the operator $S_{\mathcal{B}}$ does a phase shift of the good state

$$|x\rangle \mapsto \begin{cases} -|x\rangle, & \text{if } \mathcal{B}(x) = 1 \\ |x\rangle, & \text{if } \mathcal{B}(x) = 0 \end{cases}, \tag{3.8}$$

and S_0 changes the sign of the amplitude only for the state $|0\rangle$. After the computation of $Q^\kappa \mathcal{A}|0\rangle$, where $\kappa \approx 2^{k/2}$, the measurement yields good with probability of $1 - 2^{-k}$.

Let l denote the amount of computations run in parallel. Leander and May point out that choosing $l > 3n$, according to Kaplan et al. in [Kap+16], is sufficiently large enough. Let $h : \{0,1\}^k \times (\{0,1\}^n)^l$ be defined as the function evaluating $f(\cdot,\cdot)$ in parallel on l arguments in the second component. Thus,

$$(u, x_1, \ldots, x_l) \mapsto f(u, x_1) \| \ldots \| f(u, x_l).$$

Let the unitary operator U_h be the embedding of h on $k + 2nl$ qubits and map

$$|u, x_1, \ldots, x_l, 0, \ldots, 0)\rangle \mapsto |u, x_1, \ldots, x_l, h(u, x_1, \ldots, x_l)\rangle.$$

The quantum algorithm $\mathcal{A}$ on input $|0\rangle$ is then defined as follows

1. Start with the initial $k + 2nl$-qubit state $|0\rangle$.

2. Apply Hadamard transformation $H^{\otimes k+nl}$ on the first $k + nl$ qubits. Resulting in the following state
$$\frac{1}{\sqrt{2^{k+nl}}} \sum_{\substack{u \in \{0,1\}^k \\ x_1,\ldots,x_l \in \{0,1\}^n}} |u\rangle |x_1\rangle \ldots |x_l\rangle |0\rangle.$$

3. Apply U_h to obtain

$$\frac{1}{\sqrt{2^{k+nl}}} \sum_{\substack{u\in\{0,1\}^k \\ x_1,\dots,x_l\in\{0,1\}^n}} |u\rangle\, |x_1\rangle \dots |x_l\rangle\, |h(u,x_1,\dots,x_l)\rangle\,.$$

4. Applying Hadamard transformation again to $|x_1\rangle, \dots, |x_l\rangle$ results in

$$|\varphi\rangle = \frac{1}{\sqrt{2^{nl}}}\frac{1}{\sqrt{2^{k+nl}}} \sum_{\substack{u\in\{0,1\}^k \\ x_1,\dots,x_l\in\{0,1\}^n \\ y_1,\dots,y_l\in\{0,1\}^n}} |u\rangle\, (-1)^{y_1\cdot x_1}\, |y_1\rangle \dots (-1)^{y_l\cdot x_l}\, |y_l\rangle\, |h(u,x_1,\dots,x_l)\rangle\,.$$

Assuming that the last nl qubits of the state $|\varphi\rangle$ would be measured. These qubits would collapse to

$$|h(u,x_1,\dots,x_l)\rangle = |f(u,x_1)\,\|\dots\|\,f(u,x_l)\rangle\,,$$

for some fixed values $y, x_1, \dots, x_l$. Assume there is a κ_0, like in the previous section, for that $f(\kappa_0,\cdot)$ is periodic in s. Furthermore assume $y = \kappa_0$, then every arbitrary n-qubit state $|z_i\rangle = (-1)^{y_i\cdot x_i}\,|y_i\rangle$ would collapse into the superposition that is consistent with the measured $f(\kappa_0, x_i)$. Then x_i and $x_i \oplus s$ are the only preimages of $f(\kappa_0, x_i)$. Each $|z_i\rangle$ would collapse into the superposition

$$(-1)^{y_i\cdot x_i}\,|y_i\rangle = (-1)^{y_i\cdot x_i}(1 + (-1)^{y_i\cdot s})\,|y_i\rangle\,. \tag{3.9}$$

Like in the basic version of Simon, the amplitude is non-zero iff $y_i \cdot s = 0 \pmod 2$. Measuring yields some y_i orthogonal to s. The algorithm is successful if the l vectors $S = \{y_1, \dots, y_l\}$ span the $k-1$-dimensional space orthogonal to s.

Heuristic 2. *(Heuristic 4 in [Bon20]) Grover-meet-Simon succeeds with probability* $1 - 2^{-\alpha}$ *in* $n + \alpha/2 + 2\lceil\frac{k}{n}\rceil$ *queries per iteration. With a perfect external test, it succeeds with probability* $1 - 2^{-\alpha}$ *in* $n + \alpha/2$ *queries plus one query to the external test per iteration.*

Remark 6 in [Bon20] (Grover-meets-Simon for periodic permutation) There is a perfect test for periodic permutations which costs only 2 queries. It amounts in testing whether or not the function fulfils $f(0) = f(s')$, with s' being a guess for the correct period s. This will only be the case if the function is indeed periodic.

3.3 The offline Simon's algorithm

Until now, all applications of Simon's algorithm are in the Q2 model. Hence, the secret function has to be quantum accessible. The only known variant of Simon's algorithm not requiring quantum access is the offline Simon's algorithm presented by Bonnetain in [Bon+19]. This problem can be seen as a modification of the Grover-meets-Simon algorithm utilizes a specific structure of the periodic function. The offline Simon algorithm addresses the following problem

Constructing and finding periodic functions: Given a boolean function $g : \{0,1\}^n \to \{0,1\}^l$, the family of functions $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^l$. Assume there is a unique $\kappa_0 \in \{0,1\}^k$ such that $f(\kappa_0, \cdot)$ is periodic in s. Let P be a quantum circuit such that

$$P\,|\kappa\rangle \sum_{x\in\{0,1\}^n} |x\rangle\,|g(x)\rangle = |\kappa\rangle \sum_{x\in\{0,1\}^n} |x\rangle\,|f(\kappa, x)\rangle\,.$$

Given oracle access to g and P, find κ_0 and the period s of $f(\kappa_0, \cdot)$.

Contrary to the Grover-meets-Simon algorithm, an additional assumption is needed, that the family of functions $f(\cdot,\cdot)$ can be computed efficiently given the fixed function g. To solve the given problem, one has to take a slightly different approach. Instead of querying an oracle, the algorithm is given a database of g, a set of superpositions $\sum_x |x\rangle\,|g(x)\rangle$, as input. It then computes the periodic function from this set and is thus able to reconstruct the period s. Let $\left|\psi_g^m\right\rangle = \bigotimes_{j=1}^m \sum_x |x\rangle\,|g(x)\rangle$ denote the quantum state that contains m copies of the superpositions of the input and output of g. Similar to the Grover-meets-Simon algorithm, the state $\left|\psi_g^m\right\rangle$ is used to check if the function f is periodic, then everything is uncomputed again to get a state close to $\left|\psi_g^m\right\rangle$ again. This state can then be reused for the next iteration. This is described in the following algorithm 1 as well as in [Bon+19] and [Bon20, Algorithm 8].

Algorithm 1: The offline Simon's algorithm

```
Input: $n$, Oracle $O_g$ and a quantum Circuit $P$ that fulfills the requirements of the
       previous stated problem.
Output: $\kappa_0$
1 Query the oracle $O_g$ $m$ times and construct $\left|\psi_g^m\right\rangle$
2 amplify over $\kappa$ with the following test:
3     Compute $m$ copies of $\sum_x |x\rangle\,|f(\kappa, x)\rangle$ from $\left|\psi_g^m\right\rangle$ and $P$.
4     Apply a Hadamard gate on the first register of each copy.
5     Compute in superposition the rank $r$ of the values in each first register.
6     $b \leftarrow r \neq n$
7     Uncompute everything but the value of $b$, to recover $\left|\psi_g^m\right\rangle$
8     return $b$
```

The approach of using only classical queries is used by manually constructing the quantum superposition over g from all of the 2^n possible classical inputs. Since this can only be done in $O(2^n)$ time, queries have to be reused to have a time-efficient algorithm. The algorithm to generate $\left|\psi_g^m\right\rangle$ from the classical queries is defined in the following Algorithm 2 as well as in [Bon+19] and [Bon20, Algorithm 9]

Algorithm 2: Constructing $\left|\psi_g^m\right\rangle$ from classical queries

```
Input: $m$, Classical oracle $O_g$
Output: $\left|\psi_g^m\right\rangle$
1 $|\phi\rangle \leftarrow \bigotimes^m \sum_x |x\rangle |0\rangle$
2 for $0 \leq i < 2^n$ do
3     Query $g(i)$
4     Apply to each register in $|\phi\rangle$ the operator
          $|x\rangle |y\rangle \mapsto \begin{cases} |x\rangle |y \oplus E_k(i)\rangle, & \text{if } x = i \\ |x\rangle |y\rangle, & \text{otherwise.} \end{cases}$
5 return $|\phi\rangle$
```

Heuristic 3. *(Heuristic 5 in [Bon20]) The offline Simon's algorithm succeeds with probability* $1 - 2^{-\alpha}$ *in* $n + k + \alpha + 4$ *queries per iteration*

3.4 Simon's algorithm for truncated functions

The goal of this section is analyzing how Simon's algorithm deals with truncated 2-to-1 functions and thus additional collisions. It will be shown that these collisions are not as constraining as previously anticipated and that Simon's algorithm is still applicable. Therefore, attacks relying on Grover's algorithm to find the correct truncated value can be exponentially speed-up using the result of this section. An example for this can be seen in section 5.3. This result of this thesis enables a number of new attacks on structures which truncate their output.
This extends the results by Kaplan et al. [Kap+16] considering collisions in Simon's algorithm. In the concurrent work by Bonnetain in [Bon20], they address this issue as well. They draw similar conclusions and apply them to reversible implementations of Simon's algorithm.

A function $\mathsf{P} : \{0,1\}^n \to \{0,1\}^n$ is called periodic if there is an $0 \neq s \in \{0,1\}^n$ such that for all $x \in \{0,1\}^n$ it holds that $\mathsf{P}(x) = \mathsf{P}(x \oplus s)$. If P is a pseudorandom function, like defined in section 2.1, it is a one-to-one function and hence bijective. Therefore P can not be periodic since this would imply that it is not injective. Let $s \in \{0,1\}^n$ be fixed but arbitrary and P a pseudorandom function. The function $h : \{0,1\}^n \to \{0,1\}^n,\ x \mapsto \mathsf{P}(x) \oplus \mathsf{P}(x \oplus s)$ is periodic in precisely s. This can be easily verified since it holds that for all $x \in \{0,1\}^n$

$$h(x) = \mathsf{P}(x) \oplus \mathsf{P}(x \oplus s) = \mathsf{P}(x \oplus s \oplus s) \oplus \mathsf{P}(x \oplus s) = h(x \oplus s).$$

There can not be another element $c' \notin \{0, c\}$ that h is periodic in since that would imply that P is periodic in c' and therefore not a pseudorandom function.

From now on P defines a pseudorandom function. With the just shown observation, the function g

$$\begin{aligned} g : \{0,1\}^n &\longrightarrow \{0,1\}^n \\ x &\longmapsto \tilde{m} \oplus \mathsf{P}(x \oplus \tilde{m}) \oplus \mathsf{P}(x), \end{aligned}$$

is periodic in $s = \tilde{m}$.

Simon's algorithm subroutine is used to find a vector orthogonal to the period s. It is illustrated in Figure 3.2. After applying U_g and measuring the second register the resulting vector $g(x)$ will be called z.

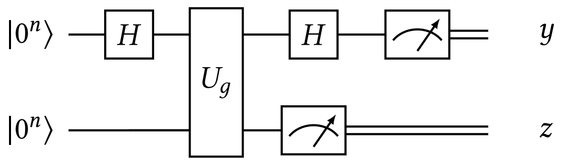


Figure 3.2: Simon's algorithm subroutine

When measuring the second register, the first register is in the state $\frac{1}{\sqrt{2}}(|x\rangle + |x \oplus s\rangle)$. Applying the second Hadamard transformation to the first register transforms it into

$$\frac{1}{2^n} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y}(1 + (-1)^{s \cdot y}) \, |y\rangle \,. \tag{3.10}$$

The amplitude $\alpha^g_{y,z}$ of y is non-zero iff $s \cdot y = 0$ and hence y is orthogonal to s.

$$\alpha^g_{z,y} = \begin{cases} \pm\frac{1}{2^{n-1}}, & s \cdot y = 0 \pmod 2, \\ 0, & s \cdot y = 1 \pmod 2. \end{cases} \tag{3.11}$$

Therefore the probability to observe y in the first register after measuring it is

$$\begin{aligned} \mathbb{P}^g[y \text{ is observed}] &= \sum_{z \in \{0,1\}^n} \|\alpha^g_{y,z}\|^2 \\ &= 2^{n-1} \left\| \frac{(1 + (-1)^{s \cdot y})}{2^n} \right\|^2 \\ &= \begin{cases} \frac{1}{2^{n-1}}, & \text{if } s \cdot y = 0 \pmod 2, \\ 0, & \text{if } s \cdot y \neq 0 \pmod 2. \end{cases} \end{aligned} \tag{3.12}$$

Thus, y_i that are not orthogonal to s are never observed. Running Simon's algorithm subroutine $O(n)$ times until $n-1$ linear independent y_i are obtained allows determining s.

2-to-1 functions that are periodic in s and truncate the output to $t < n$ bits will be considered next. The following functions O and f will be defined as

$$\begin{aligned} O : \{0,1\}^n &\to \{0,1\}^t \\ x &\mapsto \lfloor \tilde{m} \oplus \mathsf{P}(x \oplus \tilde{m}) \rfloor_t \end{aligned}$$

$$\begin{aligned} f : \{0,1\}^n &\to \{0,1\}^t \\ x &\mapsto O(x) \oplus \lfloor \mathsf{P}(x) \rfloor_t = \lfloor \tilde{m} \oplus \mathsf{P}(x \oplus \tilde{m}) \rfloor_t \oplus \lfloor \mathsf{P}(x) \rfloor_t. \end{aligned}$$

The function f which derives from g is periodic in $\tilde{m}$ as well. While each $g(x)$ has exactly two pre-images, every $f(x)$ has by far more than only the two pre-images. This holds, since the domain of f is greater than its co-domain.

One can conclude that the amount of additional collision $f(x) = f(x \oplus \zeta)$, with $\zeta \in \{0,1\}^n \backslash \{0, s\}$ is correlated to the truncated value t. Each truncated bit introduces two additional collisions, the total number is therefore 2^{n-t}. The set of additional collisions besides $\{0, s\}$ will be called χ^x and is defined as

$$\chi^x := \{\zeta \in \{0,1\}^n \backslash \{0, s\} | f(x) = f(x \oplus \zeta)\}.$$

Since f is periodic in $s = \tilde{m}$, for every collision $\zeta \in \chi^x$ there must be another $\zeta^* = \zeta \oplus s \in \chi^x$. This is important for the upcoming analysis of applying Simon's algorithm to a truncated function.

Another observation is that the set χ^x can be split into two subsets $\chi_0^x, \chi_s^x \subset \chi^x$ as follows

$$\begin{aligned} \chi_0^x &:= \{\zeta_1, \dots, \zeta_{2^{n-t-1}} \in \chi^x | \zeta_i \oplus \zeta_j \neq s, \forall\ i \neq j\}, \\ \chi_s^x &:= \chi^x \backslash \chi_0^x. \end{aligned}$$

Using Equation 3.10 and Equation 3.12 one can conclude that after measuring the second register z, the first register collapses to

$$\left((|x\rangle + |x \oplus s\rangle) + \sum_{\zeta \in \chi^x} |x \oplus \zeta\rangle \right).$$

Hence, all possible pre-images of z. After applying the second Hadamard transformation, the first register is in the state

$$\begin{aligned}
&\frac{1}{2^n}\sum_{y\in\{0,1\}^n}((-1)^{x\cdot y}+(-1)^{(x\oplus s)\cdot y}+\sum_{\zeta\in\chi^x}(-1)^{(x\oplus\zeta)\cdot y})\,|y\rangle\\
&=\frac{1}{2^n}\sum_{y\in\{0,1\}^n}((-1)^{x\cdot y}+(-1)^{(x\oplus s)\cdot y}+\sum_{\zeta\in\chi_0^x}(-1)^{(x\oplus\zeta)\cdot y}+(-1)^{(x\oplus\zeta\oplus s)\cdot y})\,|y\rangle\\
&=\frac{1}{2^n}\sum_{y\in\{0,1\}^n}((-1)^{x\cdot y}+(-1)^{(x\oplus s)\cdot y}+\sum_{\zeta\in\chi_0^x}(-1)^{(x\oplus\zeta)\cdot y}+(-1)^{(x\oplus\zeta\oplus s)\cdot y})\,|y\rangle\\
&=\frac{1}{2^n}\sum_{y\in\{0,1\}^n}((-1)^{x\cdot y}+(-1)^{(x\oplus s)\cdot y}+((-1)^{x\cdot y}+(-1)^{(x\oplus s)\cdot y})\sum_{\zeta\in\chi_0^x}(-1)^{\zeta\cdot y})\,|y\rangle\\
&=\frac{1}{2^n}\sum_{y\in\{0,1\}^n}((-1)^{x\cdot y}+(-1)^{(x\oplus s)\cdot y})(1+\sum_{\zeta\in\chi_0^x}(-1)^{\zeta\cdot y})\,|y\rangle\\
&=\frac{1}{2^n}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}(1+(-1)^{s\cdot y})(1+\sum_{\zeta\in\chi_0^x}(-1)^{\zeta\cdot y})\,|y\rangle\,.
\end{aligned}\tag{3.13}$$

As seen in Equation 3.13, the amplitude $\alpha^f_{z,y}$ is still 0 if y is not orthogonal to s. Thus,

$$\alpha^f_{z,y}=\begin{cases}\frac{2}{2^n}(-1)^{x\cdot y}(1+\sum_{\zeta\in\chi_0^x}(-1)^{\zeta\cdot y}), & \text{if } s\cdot y=0 \pmod 2,\\ 0, & \text{if } s\cdot y=1 \pmod 2.\end{cases}\tag{3.14}$$

The probability to observe such an y is

$$\begin{aligned}
\mathbb{P}^f[y\text{ is observed}] &= \sum_{z\in\{0,1\}^n}\|\alpha^f_{y,z}\|^2\\
&=\frac{1}{2^{n+1}}\left\|(-1)^{x\cdot y}(1+(-1)^{s\cdot y})(1+\sum_{\zeta\in\chi_0^x}(-1)^{\zeta\cdot y})\right\|^2\\
&=\frac{1}{2^{n+1}}\left\|(1+(-1)^{s\cdot y})(1+\sum_{\zeta\in\chi_0^x}(-1)^{\zeta\cdot y})\right\|^2\\
&=\begin{cases}\frac{1}{2^{n-1}}(1+\sum_{\zeta\in\chi_0^x}(-1)^{\zeta\cdot y})^2, & \text{if } s\cdot y=0 \pmod 2,\\ 0, & \text{if } s\cdot y=1 \pmod 2.\end{cases}
\end{aligned}\tag{3.15}$$

The last transformation of Equation 3.15 shows that the probability depends on whether ζ is orthogonal to y. However, this also shows these collisions are not arbitrary. Instead of looking at each ζ individually, the basis of χ_0^x has to be considered. This leads to the following Lemma presented by Bonnetain in [Bon20] which was adapted from [Kap+16].

Lemma 3. *(Lemma 1 in [Bon20]) Let $\mathcal{H}$ be a subgroup of $\{0,1\}^n$. Let $y \in \{0,1\}^n$. Then*

$$\sum_{h\in\mathcal{H}} (-1)^{h\cdot y} = \begin{cases} |\mathcal{H}|, & \text{if } y \in \mathcal{H}^\perp, \\ 0, & \text{otherwise.} \end{cases}$$

Proof. Let $h_1, \ldots, h_c$ be a basis of $\mathcal{H}$. Therefore, each element $h \in \mathcal{H}$ can be written as $\sum_{i=1}^{c} v_i h_i$. Thus,

$$\sum_{h\in\mathcal{H}} (-1)^{h\cdot y} = \sum_{v_1=0}^{1} (-1)^{v_1 h_1 \cdot y} \ldots \sum_{v_c=0}^{1} (-1)^{v_c h_c \cdot y} = \prod_{i=1}^{c} (1 + (-1)^{h_i \cdot y}).$$

This product is only nonzero iff each $h_i \cdot y = 0$, hence, $y \in \mathcal{H}^\perp$. In this case it is equal to $2^c = |\mathcal{H}|$. □

Using this Lemma, the Equation 3.15 can then be written as

$$\begin{aligned} \mathbb{P}^f[y \text{ is observed}] &= \begin{cases} \frac{1}{2^{n-1}}(1 + \sum_{\zeta\in\chi_0^x} (-1)^{\zeta\cdot y})^2, & \text{if } s\cdot y = 0 \pmod 2, \\ 0, & \text{if } s \cdot y = 1 \pmod 2 \end{cases} \\ &= \begin{cases} \frac{1}{2^{n-1}}(1 + |\chi_0^x|)^2, & \text{if } s\cdot y = 0 \pmod 2, \\ 0, & \text{if } s \cdot y = 1 \pmod 2). \end{cases} \end{aligned} \tag{3.16}$$

The collisions are therefore equally distributed and occur with equal probability. However, the collisions are still interfering with Simon's algorithm. After cn calls of Simon's algorithm subroutine and having vectors $y_1, \ldots, y_{cn}$ all orthogonal to s, there is the possibility that s can not be recovered. This is the case if the $(n-1)$-dimensional space orthogonal to s can not be spanned. In other words if there is a collision $\mathrm{Z} \in \{0,1\}^n$ that is orthogonal to all obtained $y_1, \ldots, y_{cn}$. The results by Kaplan et al. in [Kap+16, Section 2.2] can be applied to analyze the probability of this to happen

$$\begin{aligned} &\mathbb{P}[\dim(\mathrm{Span}(y_1, \ldots, y_{cn})) \le n-2] \\ &\quad\le \mathbb{P}[\exists\ \mathrm{Z} \in \{0,1\}^n \backslash \{0,s\} : y_1 \cdot \mathrm{Z} = \ldots = y_{cn} \cdot \mathrm{Z} = 0] \\ &\quad\le \sum_{\zeta\in\{0,1\}^n\backslash\{0,s\}} (\mathbb{P}[y_1 \cdot \mathrm{Z} = 0])^{cn} \\ &\quad\le \max_{\mathrm{Z}\in\{0,1\}^n\backslash\{0,s\}} (2\mathbb{P}[y_1\cdot \mathrm{Z} = 0]^c)^n. \end{aligned} \tag{3.17}$$

The probability is hence

$$\mathbb{P}_y[y \cdot \mathrm{Z} = 0] \le \frac{1+p_0}{2},$$

where $p_0 \ge \max_{\mathrm{Z}\in\{0,1\}^n\backslash\{0,s\}} \mathbb{P}_x[f(x) = f(x \oplus \mathrm{Z})]$.

In the specific case of the truncated function f

$$p_0 = \frac{\max_x |\chi^x|}{2^n} \approx \frac{1}{2^t}.$$

Thus, choosing $c \geq \frac{3}{1-p_0}$ ensures that the error will decrease exponentially with n. Running Simon's algorithm $O(cn)$ times ensures retrieving the correct s for truncated functions.

Bonnetain used the results by Kaplan et al. to show the following heuristic.

Heuristic 4. *(Heuristic 6 in [Bon20]) A reversible implementation of Simon's algorithm with q queries only needs a periodic function with $\lceil 3.5 + log_2(q) \rceil$ bits of output.*

3.5 Quantum attacks using Simon's algorithm

The goal of superposition attacks using Simon's algorithm, or a variant of it, is to either target a periodic function or to utilize properties of the non-periodic function to create a periodic one and and thus fulfil Simon's premise which allows the extraction of secret information. First, the application of the basic version of Simon's algorithm is considered. Here, the Even-Mansour construction is approached.

3.5.1 The Even-Mansour-construction

One of the first encryption schemes shown to be broken in the Q2 model is the Even-Mansour construction. Kuwakado and Morii presented in [KM12] an attack that only needs $O(n)$ quantum queries and operates $O(n^3)$ time, where the overhead. They also presented a key recovery attack in the Q1 model. This attack is done using $O(2^{n/3})$ classical queries and qubits, and $O(2^{n/3})$ offline quantum computations. This was optimized by Bonnetain in [Bon+19] to only use polynomially many qubits, while still requiring classical $O(2^{n/3})$ queries. This attack will be presented in subsection 3.5.4

In the classical world the Even-Mansour construction is proven to be secure [EM97]. The best known classical attack uses up to $O(2^{n/2})$ online queries and offline computations [Dae91].

Even-Mansour: The Even-Mansour construction [EM97] is a minimal block cipher. It uses two different secret keys k_0, k_1 and a publicly known permutation P. The two secret keys are used as whitening keys. After the first whitening with k_0, the publicly known permutation is applied and the permuted result is then XORed with the k_1. It is proven to be secure in the classical world, where P is chosen randomly amongst all possible permutations.

This construction is proven to be unsecure in the quantum world where an adversary has access to an quantum encryption oracle. Kuwakado and Morii showed in [KM12] that an adversary can query the encryption oracle to receive $Enc_{EM}(x) = P(x \oplus k_0) \oplus k_1$ and since the permutation P is publicly known, the adversary is able to build following function

$$f(x) = Enc_{EM}(x) \oplus P(x) = P(x \oplus k_0) \oplus k_1 \oplus P(x),$$

which fulfills $f(x) = f(x \oplus k_0)$ and thus is periodic in k_0. By using the basic version of Simon's algorithm, the adversary is able to extract the secret k_0 in linear time and hence getting k_1 by querying a single classical query to an encryption oracle.

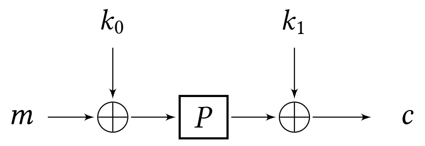


Figure 3.3: The Even-Mansour construction

3.5.2 The three-round-Feistel-construction

Another application of Simon's algorithm was discovered by Kuwakadi and Mori as well. This time the three-round-Feistel construction was broken in the Q2 model. They presented their work in [KM10]. Inspired by their attack, Santoli and Schaffner presented a new and optimized version of this attack in [SS16]. An adversary is able to build a quantum distinguisher $\mathcal{D}$ that is able to efficiently differentiate between a three-round-Feistel network and a random permutation.

Let F_i with $i = 1, \dots, 3$ denote the internal round functions of the three-round-Feistel scheme with input (L_0, R_0) and output $(L_3, R_3) = E(L_0, R_0)$. An illustration of the three-round-Feistel scheme is presented in Figure 3.4 below.

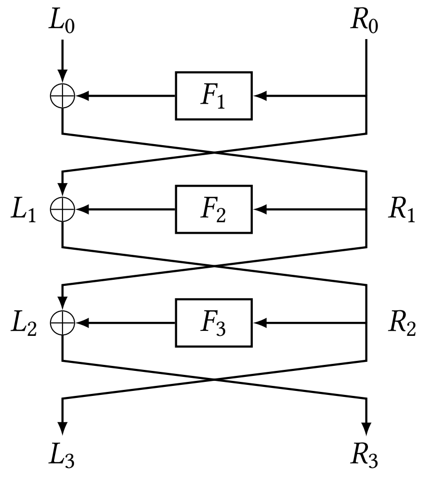


Figure 3.4: The three-round-Feistel scheme.

The quantum distinguisher $\mathcal{D}$ presented by Santoli and Schaffner is able to distinguish between the three-round-Feistel scheme and a random permutation using the first n bits of the $2n$ permutation. Looking at Figure 3.4, one can see that these first n bit can be defined as $L_3 = R_0 \oplus F_2(L_0 \oplus F_1(R_0))$, if the oracle O implemented the Feistel scheme. Otherwise L_3 will be a random n-bit string. The first n-bit of a $2n$-bit string S will henceforth be defined as $left(S)$.
Let $\alpha_0 \neq \alpha_1 \in \{0,1\}^n$ be two arbitrary constants then the function f can defined as follows

$$\begin{aligned} f : \{0,1\} \times \{0,1\}^n &\rightarrow \{0,1\}^n \\ (b, x) &\mapsto left(O(x, \alpha_b)) \oplus \alpha_b. \end{aligned} \tag{3.18}$$

If the oracle O implemented the three-round-Feistel scheme, f can be written as $f(b,x) = F_2(x \oplus F_1(\alpha_b))$ and f satisfies

$$f(b,x) = f(b \oplus 1, x \oplus R_1(\alpha_0) \oplus R_1(\alpha_1)).$$

Furthermore,

$$\begin{aligned} f(b',x') = f(b,x) &\Leftrightarrow x' \oplus F_1(\alpha_{b'} = x \oplus F_1(\alpha_b) \\ &\Leftrightarrow \begin{cases} x' \oplus x = 0, & \text{if } b' = b, \\ x' \oplus x = F_1(\alpha_0) \oplus F_1(\alpha_1), & \text{if } b' \neq b. \end{cases} \end{aligned} \tag{3.19}$$

The function is periodic in $s = 1 \,||\, F_1(\alpha_0) \oplus F_1(\alpha_1)$ and fulfills Simon's promise. After at most $2n$ iterations of Simon's algorithm the distinguisher $\mathcal{D}$ is able to recover the period s if the oracle implemented the Feistel structure, otherwise the distinguisher can assume that the oracle implemented a random permutation.

3.5.3 The FX-construction

The FX-construction, presented by Killian and Rogaway in [KR96], is similar to the Even-Mansour construction. Two secret keys k_0 and k_1 are used as pre- and post-whitening keys. Instead of using a random permutation, a secure block cipher E using a secret key k_2 is used instead. This construction is considered secure in the classical world if the block cipher itself is considered secure.

The previously presented quantum attack on the Even-Mansour construction is not applicable here, since Simon's Algorithm can only be used if $Enc_{FX}(x) \oplus E_{k_2}(x)$ is periodic. This is only the case if the adversary is able to guess k_2 correctly.

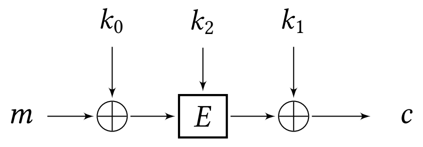


Figure 3.5: The FX construction.

However, Leander and May have shown in [LM17] that they are able to break the FX-construction in a quantum setting. Like previously stated, the function $f(k,x) = Enc_{FX}(x) \oplus E_k(x) = E_{k_2}(x \oplus k_0) \oplus k_1 \oplus E_k(x)$ is periodic in k_1 iff $k = k_0$. Using the Grover-meets-Simon algorithm (see section 3.2) one is able to find the correct k. Having access to an encryption oracle O implementing Enc_{FX} an adversary is able to recover k_0, k_1, k_2 by using $m + 4n(n + \sqrt{n})$ qubits and making $\mathcal{O}(m+n) \cdot 2^{m/2}$ queries to a quantum oracle (see [LM17, Section 3, Theorem 2]), where n is the length of k_0, k_1 and m the length of k_2. This is comparable with to the attack without using whitening keys. Therefore they came to the conclusion that using whitening keys does not help against quantum adversaries.

3.5.4 Breaking Even-Mansour using the offline Simon's algorithm

The attack using the basic version of Simon's algorithm in the Q2 model was presented earlier. Not having access to a quantum oracle and thus the secret keys k_0 and k_1 in superposition makes the attack more difficult. The base idea behind this attack is dividing the $n-$bit secret key k_0 into two subkeys $k_0^{(1)}$ of u-bits and $k_0^{(2)}$ of $n-u$-bits. Simon's algorithm is then applied to recover $k_0^{(1)}$ while guessing $k_0^{(2)}$ using Grover's algorithm.

For an integer u such that $0 \leq u \leq n$ define the function

$$\begin{aligned} F : \{0,1\}^{n-u} \times \{0,1\}^{u} &\to \{0,1\}^{n}, & g : \{0,1\}^{u} &\to \{0,1\}^{n} \\ i, x &\mapsto P(x \,\|\, i), & x &\mapsto \mathtt{tag}_1(x \,\|\, 0^{n-u}). \end{aligned} \tag{3.20}$$

The subkey $k_0^{(2)}$ can then be recovered by applying the offline Simon's algorithm presented in section 3.3 together with F and g. Since $k_0^{(1)}$ is the period of $F(k_0^{(2)}, x) \oplus g(x)$ it can be therefore recovered with the just found $k_0^{(2)}$. The second secret key k_1 can then be computed with a classical query to the oracle knowing k_0.

4 Quantum error correction

When considering building a quantum computer, quantum error correction is an essential part. Quantum states are fragile and sensitive to noise. Protecting quantum information against the effect of noise is therefore very important. Contrary to the classical counterpart the bit, there is more than just one type of error than can occur. Indeed, there are an infinite number of errors that can occur. Developing a quantum error correcting code (QECC) and therefore a fault tolerant quantum computer seems therefore to be impossible. However, this chapter will show that it is nonetheless possible to create such codes that can not only detect but also correct any arbitrary quantum error there is.

A QECC is defined as a subspace of the Hilbert space designed such that any of a set of possible errors can be corrected by an specific quantum operation [Got06]. An example for this will be given in form of Shor's 9-qubit code to show that QECC exist and are possible. Afterwards the formalism of stabilizer are defined and used to introduce the family of topological stabilizer codes called the surface codes. The fault-tolerant quantum computer used for the cost estimations in chapter 6 is assumed to be be based on surface codes.

The results of this chapter are based on the work of Gottesman in [Got06], Nielsen and Chuang in [NC01], Fowler et al in [Fow+12], Terhal in [Ter15], Devitt et al. in[DMN13], and Todd in [Bru20].

4.1 Introduction

The simplest way to protect classical bits against errors is to use redundancy. A bit, represented by 0 and 1, is encoded as multiple bits by copying or repeating it.

$$0 \mapsto 000 \text{ and } 1 \mapsto 111.$$

A single error can then be detected and corrected by taking the majority of the three bits. This type of decoding is called majority voting or repetition code. The probability that two or more bits were flipped and thus an error would not be detected or corrected is $3p^2 - 2p^3$, assuming a single bit-flip occurs with probability p. There are of course more efficient procedures, but the key idea is always to encode messages by adding enough redundancy. [NC01]

For the question whether this technique was applicable for quantum error correction, the answer is no. A quantum state can not be copied due to the no-cloning theorem, see Theorem 1.

To protect quantum states and therefore develop quantum error-correcting codes, some barriers have to be overcome first. In particular, there are four difficulties to deal with.

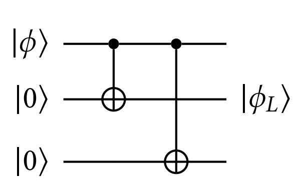
1. No-cloning theorem forbids repetition.
2. Measurement of error destroys quantum information: In classical error correction the output is observed to determine whether an error occurred and what decoding procedure to adapt. In quantum mechanics measuring a state destroys its superposition and makes recovery impossible.
3. Must correct multiple types of errors: Unlike classical bits, qubits can also have phase errors in addition to bit-flip errors.
4. Errors are continuous: The rotation, or the phase factor, of an phase error can have arbitrary angle θ. Therefore quantum errors are continuous and an infinite amount of different errors exist.

Even though the classical repetition code approach can not be recreated for qubits, its idea can still be applied in some way.

3-qubit error correction code Due to the no-cloning theorem the state $|\phi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ can not be copied to $|\phi\rangle^{\otimes 3}$. Instead, the base states $|0\rangle$ and $|1\rangle$ can be encoded in a convenient way to

$$\begin{aligned} |0\rangle &\rightarrow |0_L\rangle = |000\rangle, \\ |1\rangle &\rightarrow |1_L\rangle = |111\rangle, \end{aligned} \tag{4.1}$$

where $|0_L\rangle$ and $|1_L\rangle$ define the logical $|0\rangle$ and $|1\rangle$ state and not the physical one. The quantum state$|\phi\rangle$ is then encoded as the 3-qubit codeword $|\phi_L\rangle = \alpha_0 |0_L\rangle + \alpha_1 |1_L\rangle$. The creation of this state can be seen in Figure 4.1.



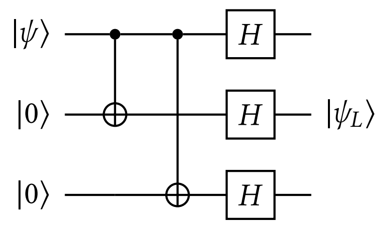


Figure 4.1: Creating the states $|\phi_L\rangle = \alpha_0 |000\rangle + |111\rangle$ and $|\psi_L\rangle = \alpha_0 |+++\rangle + \alpha_1 |---\rangle$

This code is able to correct a single bit-flip error that occurred on one or fewer qubits. A bit flip is equivalent to applying an X operator to a qubit, since $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. Therefore, applying an X operator to the flipped qubit corrects it.
To get the correct position of the flipped qubit, two additional ancilla qubits are introduced. Instead of measuring $|\phi_L\rangle$ and thus destroying its superposition, it is possible to extract information about the error without revealing information about $|\phi_L\rangle$. The first ancilla qubit checks the parity of the first two qubits of $|\phi\rangle$ and the second ancilla checks the last two qubits. This is illustrated in Figure 4.2. The measurement on the error syndrome leaves the state unchanged and the error syndrome only contains information about what error has occurred and at what position. The classical error syndrome result then tells what action has to be taken to recover the initial state.

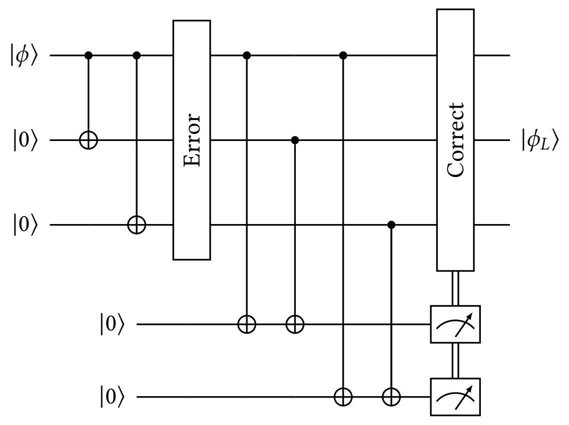


Figure 4.2: Encoding and correcting single bit-flip error using two ancilla qubits. [DMN13]

Error location	Ancilla measurement	Correction
no error	00	nothing
Qubit one	11	Apply σ_x on qubit one
Qubit two	10	Apply σ_x on qubit two
Qubit three	01	Apply σ_x on qubit three

Table 4.1: Ancilla measuring result for single bit-flip error.

The error correction works if at most a single bit-flip occurred on the three qubits. With this 3-qubit code it is possible to correct a quantum bit-flip error without measuring and without copying a state.

Next, phase-flip errors will be considered. A phase-flip occurs when a phase factor of -1 is applied between the two base states. This can be expressed with the Pauli Z operator.

$$|\phi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle \rightarrow Z |\phi\rangle = \alpha_0 |0\rangle - \alpha_1 |1\rangle . \tag{4.2}$$

The Hadamard transformation converts between the $|0\rangle, |1\rangle$ basis and the $|+\rangle, |-\rangle$ basis where the X operator acts like a phase-flip and the Z operator like a bit-flip. Therefore, the just presented 3-qubit code to correct a bit-flip can be used in the $|+\rangle, |-\rangle$ basis to correct a phase-flip. The quantum circuit used for encoding the codeword $|\psi_L\rangle = \alpha_0 |+++\rangle + \alpha_1 |---\rangle$ can be seen on the right of Figure 4.1. Detecting and correcting phase-flips errors using this code functions exactly like the bit-flip code with the basis change noted above.

Both the bit-flip code and the phase-flip code are able to correct a finite set of errors without copying or measuring the state. But they are only able to correct their own set of errors and not the other's and therefore do not represent a full quantum error correcting code. However, combining both of these codes allows to correct phase- and bit-flips errors in one code.

4.2 Shor's 9-qubit code

In 1995 Peter Shor presented in [Sho95] a quantum error correcting code utilizing nine physical qubits to encode a single logical qubit, capable of correcting any arbitrary error. The Shor code is a combination of the 3-qubit bit-flip code and the 3-qubit phase-flip code. The qubit is first encoded using the phase-flip code and then using the bit-flip code, this process is shown in Figure 4.3. Hence, encoding the codeword $|\varphi_L\rangle = \alpha_0 |0_L\rangle + \alpha_1 |1_L\rangle$ is

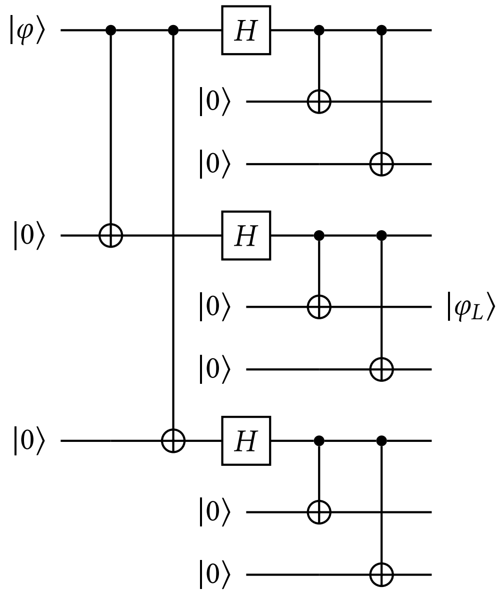


Figure 4.3: Quantum circuit used for encoding the Shor 9-qubit code.

defined by the code base states

$$\begin{aligned}
|0\rangle \longrightarrow |0_L\rangle &= \frac{1}{\sqrt{8}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle), \\
|1\rangle \longrightarrow |1_L\rangle &= \frac{1}{\sqrt{8}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle).
\end{aligned} \tag{4.3}$$

Correcting a bit-flip error is no problem since each of the 3 qubits encoded to $(|000\rangle + |111\rangle)/\sqrt{2}$ blocks is a 3-qubit bit-flip code. The error can then be corrected using the quantum circuit shown in Figure 4.1. Phase-flip errors will change the sign in one block $(|000\rangle \pm |111\rangle)/\sqrt{2} \longrightarrow (|000\rangle \mp |111\rangle)/\sqrt{2}$. Similar to the 3-qubit phase-flip code the error syndrome measurement begins by comparing the sign of the first and the second blocks. If a phase flip occurred the signs will be different and by comparing the second and third blocks the error can be detected and corrected.

Therefore, Shor's code can detect and correct any bit-flip (X) and phase-flip (Z) errors. Suppose a phase-flip and bit-flip error XZ occurred on the same qubit. Applying Shor's code to first detect and correct the bit-flip and afterwards correct the phase-flip will correct the XZ errors as well. Since it holds that $Y = iXZ$ the code corrects Y errors and therefore corrects any Pauli error acting on a single qubit. As seen in Equation 2.2 any single-qubit operator F, represented by a complex 2×2 matrix, can be written as a linear combination of the Pauli operators X, Y, Z (and I)

$$F = \epsilon_0 I + \epsilon_1 X + \epsilon_2 XZ + \epsilon_3 Z, \tag{4.4}$$

with complex numbers $\epsilon_0, \epsilon_1, \epsilon_2$ and ϵ_3. The quantum state $F \left|\varphi_L\right\rangle$ produces a superposition of four orthogonal states: $\left|\varphi_L\right\rangle$ with no error, $X\left|\varphi_L\right\rangle$ the state with an single X error, $Z\left|\varphi_L\right\rangle$ the state with a single Z error and $XZ\left|\varphi_L\right\rangle$ the state with both an X and Z error. Measuring the error syndrome collapses the superposition into one of these states and then the corresponding correction can be applied that will return the state to $\left|\varphi_L\right\rangle$. This results in the following theorems [Got06].

Theorem 4. *A quantum error correcting code (QECC) that corrects errors E_A and E_B will also correct $\alpha E_A + \beta E_B$ using the same recovery operations.*
Every QECC that corrects the single-qubit errors X, Y, Z (and I) corrects every single-qubit error. Correcting all $t-$qubit errors X, Y, Z (and I) on $t-$qubits corrects all $t-$qubit errors.

Therefore, quantum error correction is possible.

4.3 Stabilizer

The previous section showed that errors can occur as random appearances of X and Z operations. One way that they can be detected is by constantly performing X and Y measurements [Fow+12]. This can be done with a combined XZ measurement but since X and Z anticommute, as seen in section 2.2, sequential measurements would destroy the original state. No state has a eigenstate of both X and Z.

This projective measurement problem can be solved by measuring more than one qubit and therefore making non-destructive error quantum error detection possible. Consider having two qubits a and b instead. The two qubit operators X_aX_b and Z_aZ_b commute even tough they represent X and Z measurements [Fow+12, Section II].

$$\begin{aligned}
[X_aX_b, Z_aZ_b] &= (X_aX_b)(Z_aZ_b) - (Z_aZ_b)(X_aX_b) \\
&= X_aZ_aX_bZ_b - Z_aX_aZ_bX_b \\
&= (-Z_aX_a)(-Z_bX_b) - Z_aX_aZ_bX_b \\
&= 0
\end{aligned} \tag{4.5}$$

This is due to the fact that operators on different qubits always commute. Measurements of the operators are therefore compatible and a two-qubit state $|\psi\rangle$can actually be in the same eigenstate of both X_aX_b and Z_aZ_b. These two-qubit states $|\psi\rangle$ are the so called Bell-states or EPR-pairs, see section 2.2. The corresponding eigenstates of X_aX_b and Z_aZ_b as well as the Bell-states are displayed in Table 4.2 below.

Z_aZ_b	X_aX_b	$\lvert\psi\rangle$
+1	+1	$\frac{1}{\sqrt{2}}(\lvert 00\rangle + \lvert 11\rangle)$
+1	-1	$\frac{1}{\sqrt{2}}(\lvert 00\rangle - \lvert 11\rangle)$
-1	+1	$\frac{1}{\sqrt{2}}(\lvert 01\rangle + \lvert 10\rangle)$
-1	-1	$\frac{1}{\sqrt{2}}(\lvert 01\rangle - \lvert 10\rangle)$

Table 4.2: The Eigenstates of Z_aZ_b, X_aX_b, and Bell-states $\lvert\psi\rangle$

The operators Z_aZ_b and X_aX_b are called stabilizers. Using the formalism introduced by Gottesman in [Got98a] a state $\lvert\phi\rangle$ is defined to be stabilized by the operator M if it is a +1 eigenstate of M: $M\lvert\phi\rangle = \lvert\phi\rangle$. The group of stabilizers S of a code is Abelian and therefore commute with one another, a subgroup of the Pauli group P_N and define the codespace $T(S) = \{\lvert\phi\rangle\, s.t. M\lvert\phi\rangle = \lvert\phi\rangle\, \forall M \in S\}$. For example, the 3-qubit code has the stabilizers $S = \{I_1I_2I_3, Z_1Z_2I_3, I_1Z_2Z_3, Z_1I_2Z_3\}$. The parity check on the first and second qubit can be seen as the projective error measurement using the stabilizer $Z_1Z_2I_3 = Z_1Z_2$ and on the second and third qubits as Z_2Z_3. These two elements are the generators of the stabilizer code, here the 3-qubit bit-flip code. The phase-flip code has the generators X_1X_2 and X_2X_3. Shor's code is generated by the stabilizers

$$\begin{aligned} S = \{&Z_1Z_2, Z_2Z_3, Z_4Z_5,\\ &Z_5Z_6, Z_7Z_8, Z_8Z_9,\\ &X_1X_2X_3X_4X_5X_6,\\ &X_4X_5X_6X_7X_8X_9\} \end{aligned}$$

Measuring the quantum states with a set of commuting stabilizers forces the state into a simultaneous eigenstate of all stabilizers. Using them to measure errors will therefore not alter the state $\lvert\psi\rangle$. The quantum state is instead projected into a different stabilizer eigenstate if an error occurred [Fow+12, Section III].

4.4 Fault-tolerant quantum computations

Fault-tolerant quantum computation can be defined as a set of principles that enable error correction during quantum computation without more errors occurring than are corrected. Or in other words to provide reliable quantum computations when the basic components are not reliable. Brun gives in [Bru20] a good overview of the principles of fault tolerance.

1. Never decode the quantum information. All operations must be done on the encoded quantum data.

2. Quantum circuits acting on encoded data should be robust against errors. The circuits should not cause a correctable error to spread until it is an uncorrectable error.

3. The encoded information should be corrected periodically, to catch and remove errors before they accumulate.

4. Error correction circuits also should not spread errors.

5. It is impossible ever to remove all errors; but any residual errors should be correctable, so they can be caught and removed in the next error-correction step.

The principle to not cause error propagation is very important especially for multi-qubit gates. Looking at the CNOT operation between two qubits, even if the gate itself is perfect it can propagate errors forward. Suppose, the qubits are in the state $|0\rangle\,|0\rangle$ an error occurs and flips the bit of the first qubit changing it to $|1\rangle$. The CNOT will then propagate this error forward and thus the second qubit is also erroneous [Got06]. This process is illustrated in Figure 4.4 below.

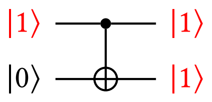


Figure 4.4: Error propagation of the CNOT gate

The solution for this problem is to use transversal gates. Therefore, an operation in which only the ith qubit in each block interacts with the ith qubit of the other blocks of the code or ancilla states. A qubit error in any block, assuming the 1st block, can thus only propagate to the same qubit,so the first qubit, of the other blocks of the code [Got06].

Fault tolerant quantum computing begins therefore with a quantum error correcting code and a universal set of gates. These gates should be fault-tolerant. As mentioned in section 2.2 the Clifford+T gate set is an example for an universal set of gates. Therefore, a common approach on fault tolerant quantum computing is to use a QECC that allows efficient encoded Clifford gates and additionally use the magic state distillation process to provide fault-tolerant T-gates.

An important aspect of fault-tolerant quantum computation is the threshold theorem. It states that if the error rate per gate is below a certain error threshold, then it is possible to perform arbitrary long quantum computations [Got06; Fow+12; Jon+12]. The threshold is dependant on the code being used and topological codes like the surface codes have one of the highest fault-tolerant thresholds in the order of $\mathcal{O}(10^{-2} - 10^{-3}$ [Wan+09].

The requirements for fault-tolerance are therefore having low gate error rates, being able to perform quantum operations in parallel, errors are not strongly correlated across the qubits, and providing new initialized qubits during quantum computation [Got06; Bru20].

4.5 Surface codes

The surface code is a stabilizer code defined over a two-dimensional qubit lattice of physical qubits. The distance d of the surface code is defined as the size of the lattice. As mentioned in the previous section, the surface code has one of the highest fault-tolerant thresholds but needs in the order of 10^4 up to 10^5 physical qubits to encode a single logical qubit, depending on the threshold and surface code.

There are two different types of physical qubits on the lattice. Data qubits in which quantum information is stored and adjacent to each data qubits are four measuring qubits, illustrated in Figure 4.5(a). The measurement qubits are used to stabilize and manipulate the quantum state of the data qubits. They consist of two different types of measurement qubits, the measure-X qubits, which are colored yellow in Figure 4.5(a), and the measure-Z qubits colored green. Each of the data qubits is adjacent to two measure-X and two measure$Z$ qubits and each measure qubit is adjacent to four data qubits.

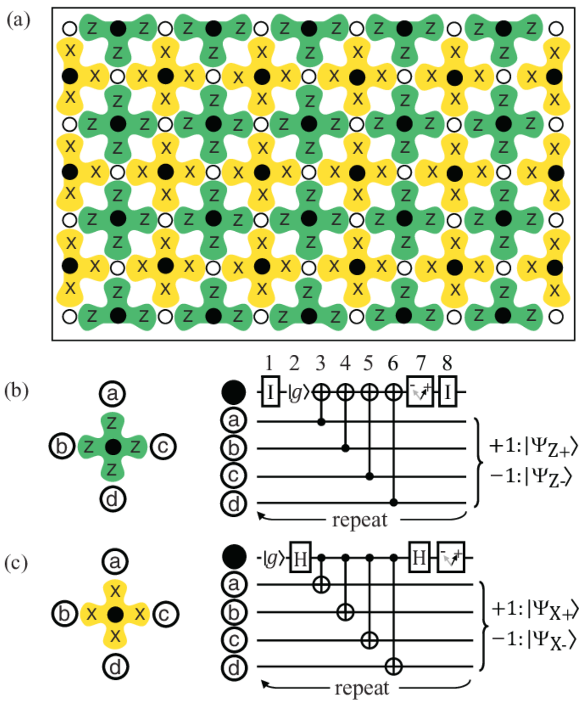


Figure 4.5: (a) Two dimensional lattice implementation of the surface code. Data qubits are open circles and measurement qubits filled circles. The measure qubits for the Z-syndrome are colored green and for the X-syndrome colored yellow. (b) measure−Z and (c) measure−X qubit measurement circuit. Figure 1 in [Fow+12] by Fowler et al.

The measure-X forces its neighbour data qubits a, b, c, and d into an eigenstate of the operator product $X_aX_bX_cX_d$ and thus measure a X stabilizer. It can be used to detect phase-flips. The measure-Z qubits measure a Z stabilizer $Z_aZ_bZ_cZ_d$ and detect bit-flips. The $Z_aZ_bZ_cZ_d$ and $X_aX_bX_cX_d$ stabilizer commute, which can be shown similar to the example in Equation 4.5.

The measure-X and measure-Z qubits stabilize the surface code and their quantum circuit can be seen in Figure 4.5(b)-(c). After the measurement by all measurement qubits in the lattice, the state of all data qubits $|\phi\rangle$ is simultaneously in $Z_aZ_bZ_cZ_d\,|\phi\rangle = Z_{abcd}\,|\phi\rangle$ with eigenvalues $Z_{abcd} = \pm 1$ and $X_aX_bX_cX_d\,|\phi\rangle = X_{abcd}\,|\phi\rangle$ with eigenvalues $X_{abcd} = \pm 1$. The measure-X or -Z qubits detect an error indicated by unwanted changes in the outcome of the measurement. It is possible to locate the exact physical qubit on which the error occurred since every data qubit is adjacent to two measure-X and -Z qubits. A phase- or bit-flip will be picked up by a change in the sign of the eigenvalue of the corresponding measure-X or -Z qubits and the error can be corrected. A $Y = XZ$ error can be detected by the sign changes of both the measure-X and -Z qubits adjacent to the data qubit.

Since the surface code is able to correct and detect unwanted X,Z, and XZ errors it can therefore detect and correct any arbitrary quantum error as seen in Theorem 4. These errors are not corrected by any quantum process but handled by a classical control computer. The software notes where the error occurred and simply changes the sign for future measurement outcomes of the two adjacent measure-X or -Z qubits depending on the error. This technique is called the Pauli-frame [Fow+12; Jon+12].

4.5.1 Magic state distillation

One important aspect of the surface code is providing high fidelity logical implementations of the phase gate S and the T-gate. Both rely on special ancilla states

$$|A_L\rangle = \frac{1}{\sqrt{2}}\left(|0_L\rangle + e^{i\pi/4}|1_L\rangle\right) \text{ and } |Y_L\rangle = \frac{1}{\sqrt{2}}\left(|0_L\rangle + i|1_L\rangle\right). \tag{4.6}$$

The ancilla state $|Y_L\rangle$ is used for implementing a logical S-gate and $|A_L\rangle$ for implementing a logical T-gate. The process for creating the $|Y_L\rangle$ and $|A_L\rangle$ states are quite similar, therefore only the magic state $|A_L\rangle$ will be considered, since it is more important for the error cost analyses in chapter 6.

The so called magic states are created by injecting specific qubits with an imperfect state and then purified them by using a process called distillation. This process is repeated until the desired fidelity of the magic states is achieved. Afterwards the T_L-gate can be implemented using the logical circuit illustrated in Figure 4.6. The ancilla state $|A_L\rangle$ is used to control the CNOT on the target state $|\psi_L\rangle$. A projective measurement onto a Z stabilizer is done resulting in a probabilistic outcome. If the outcome is 1 the state collapses to the correct state $|\phi_L\rangle = \alpha_0|0\rangle + e^{i\pi/4}|1\rangle$. If the measuring outcome is -1 the qubit is in the state $|phi_L\rangle = XT_L^\dagger$. The correction of the state is achieved by applying a S_L gate to $|\phi_L\rangle$. This will result in $S_L|\phi_L\rangle = (X_L Z_L)T_L|\psi_L\rangle$. The $X_L Z_L$ are byproduct operators that can be handled by the control software, applying the Gottesman-Knill Theorem 2 that Clifford group gates can be simulated using classical computer.

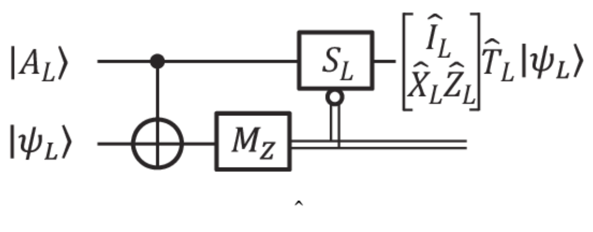

$|A_L\rangle$
$|\psi_L\rangle$
S_L
M_Z
$\begin{bmatrix}\hat{I}_L \\ \hat{X}_L\hat{Z}_L\end{bmatrix}\hat{T}_L|\psi_L\rangle$

Figure 4.6: Logic circuit for the T_L-gate. Figure 30 in [Fow+12] by Fowler et al.

The magic state distillation process is probabilistic. The imperfect state is purified by repeating a specific logic circuit. After each round of execution, the output state approaches the desired result rapidly. For the $|A_L\rangle$ ancilla states, the 15-qubit Reed-Muller encoding presented in [Fow+12; RHG06; BK05b], is used. The used logical circuit is shown in Figure 4.7 and can be implemented by the surface code very efficiently.

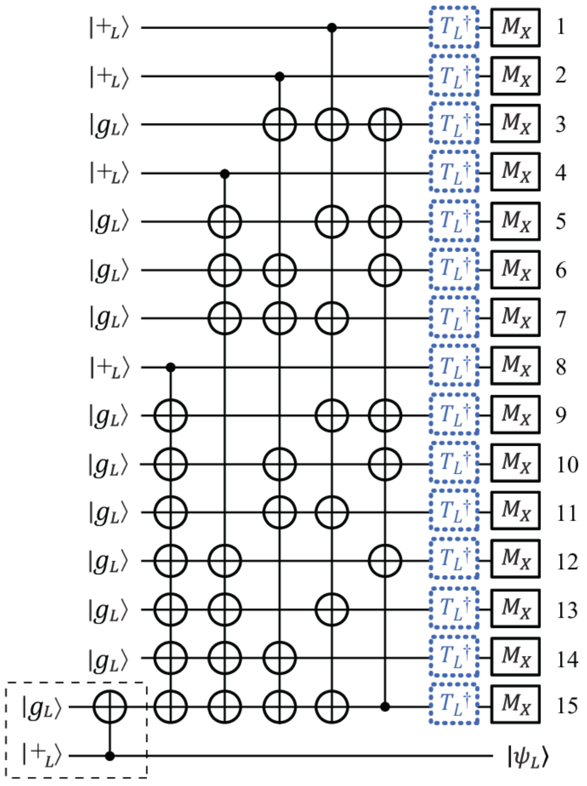


Figure 4.7: Distillation circuit for the $|A_L\rangle$ ancilla state. Figure 33 in [Fow+12] by Fowler et al.

In the shown circuit, a logical Bell pair is created and one of the logical qubits from the pair is entangled with 14 other logical qubits. The 15 qubits are then rotated with the help of approximated $T_L^\dagger$-gates, where each gate uses an approximation of the $|A_L\rangle$ state as an ancilla. In These ancillas are either obtained by a state injection in the first round or later from the previous round of distillation. The distillation circuit then generates one output state $|\psi_L\rangle$ in the other logical qubit of the Bell-pair. Afterwards 15 projective measurements into X-stabilizers are made to evaluate the four stabilizers for the Reed-Muller code

$$\begin{aligned}
X_{R_1} &= X_{L4}X_{L5}\dots X_{L11} \\
X_{R_2} &= X_{L1}X_{L2}\dots X_{L7}X_{L15} \\
X_{R_3} &= X_{L2}\dots X_{L5}X_{10}\dots X_{L13} \\
X_{R_4} &= X_1X_2X_5X_6X_9X_{10}X_{13}X_{14}.
\end{aligned} \tag{4.7}$$

If each of the stabilizers measurement outcomes is $\{+1, +1, +1, +1\}$, the state of the other bell-pair qubit $|\psi_L\rangle$ is a purified $|A_L\rangle$ state.

Given an input error rate p_{in}, the output state $|\psi_L\rangle$ will have a new error rate of $35p_{in}^3$ upon successfully distilling an $|A_L\rangle$ state. The rate at which the magic state distillation process fails is $1 - 15p_{in}$.

5 Superposition attacks on lightweight cryptography

This chapter presents various attacks on lightweight cryptographic schemes. These include two candidates of the Lightweight Cryptography Standardization Process introduced by the National Institute of Standards and Technology, namely the finalist `Elephant` and the second-round candidate `ESTATE` . The third introduced attack is on the lightweight scheme `LightMAC` . All attacks use variations of Simon's algorithm presented in chapter 3.

All three attacks against `Elephant` are key recovery attacks and the result of this thesis. The attacks against `Elephant` cover both the second-round submission as well as the final-round submission of the Lightweight Standardization Process. The presented attacks are in both the Q2 as well as the Q1 model.

The attacks on `ESTATE` and `LightMAC` are from Jonas Haas' master thesis [Haa20], who presented a distinction attack on the pseudorandom permutation of `LightMAC` as well as a tag-forgery attack on `ESTATE` . Both presented attacks are in the Q2 model. The results of this thesis in section 3.4 can be applied to the attack on `LightMAC` , reducing its cost exponentially.

5.1 Lightweight cryptography

Lightweight cryptography is a class of modern cryptography that includes cryptographic algorithms designed for use in devices with low resources [PS11]. The use of small computing devices like radio frequency identification (RFID) tags, Internet of things (IoT) devices, measuring devices, and wireless sensors is becoming more and more common in the day-to-day activity. The move from personal computers to small devices brings with it a whole new set of security and privacy issues. It is therefore difficult to balance the trade off between security and performance of resource restricted devices [Bas+18]. In 2013, the National Institute of Standards and Technology (`NIST`) launched the Lightweight Cryptography Standardization Process to study and evaluate the performance of current cryptography standards on constrained devices. The goal of this project is to standardize one or more Authenticated Encryption with Associated Data (AEAD) and hashing schemes that are suitable for use in constrained environments as in the examples given above [Sön+21]. In response to its first call for algorithms `NIST` received 57 submissions of which 56 were accepted and announced as first-round candidates. In August 2019, the second-round candidates were announced. From the initial 56 submissions 32 made it to the second round. About a year later, in August 2020, the submitters of the second-round candidates were asked to provide an update of their algorithms in particular on new proofs that support the security claims, new software and hardware implementations, and other claims that are listed in [Sön+21]. In March 2021, NIST announced the finalist of the Lightweight Cryptography Standardization Process.

5.1.1 NIST lightweight standardization finalists

The finalists of the Lightweight Cryptography Standardization Process are

- ASCON
- Elephant
- SPARKLE
- Romulus
- GIFT-COFB
- PHOTON-Beetle
- Grain-128AEAD
- TinyJAMBU
- ISAP
- Xoodyak

The cryptographic security of the candidates is the most important criteria for evaluation. Submissions with significant third-party analysis or security claims based on well-understood design principles and security proofs were favored for the finalists selection. The second criterion was the performance of the candidates in constrained environments. Another criterion of the candidates was the side-channel resistance. [Sön+21]

5.2 Superposition attacks on Elephant

Elephant was first proposed by Mennink et al. in 2019 [Men21]. It is a family of authenticated encryption schemes. It is currently one of the ten NIST lightweight standardization finalists, see subsection 5.1.1. The mode of Elephant is a permutation- and nonce-based encrypt-then-MAC construction. The message authenticated and encryption mode both use a cryptographic permutation internally which is masked using LFSRs, like the masked Even-Mansour construction of Granger et al. [Gra+16a]. The permutation is only evaluated in the forward direction and hence inverse-free.

Two versions of Elephant will be presented here. Elephant v1.1 [DM19] was presented in 2019 for the second round of the NIST lightweight standardization challenge and v2 [Men21] was introduced in 2021 as a candidate of the final round of said NIST challenge. In both versions are vulnerabilities which allows an adversary to recover the encryption key in the Q2 model. Both attacks will be presented in subsection 5.2.2 and subsection 5.2.3. The differences between the two versions are minor. Hence only the specifications of Elephant v2 will be introduced in the following section. All changes of the Elephant v1.1 will be noted in subsection 5.2.3.

5.2.1 Specification

The Elephant authenticated mode consists of two algorithms, encryption enc and decryption dec. The description of the decryption algorithm is not important for the attacks and can be found in [Men21].

Let $k, m, n, t \in \mathbb{N}$ with $k, m, t \leq n$. The encryption algorithm enc takes a key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^m$, associated data $A \in \{0,1\}^*$, and a message $M \in \{0,1\}^*$ as input. It outputs a ciphertext $C \in \{0,1\}^{|M|}$ and a tag $T \in \{0,1\}^t$. The description of the algorithm enc is given in Algorithm 3. The size of the key K is always 128-bit and the nonces are restricted to 96-bit. The only tunable parameter of Elephant is $t \in \mathbb{N}$ which defines the length of the tag T.
The algorithm enc consists of two parts. The first four lines of Algorithm 3 describe the generation of the ciphertext C which will be outlined later in this section. The second part of the algorithm describes the authentication part with additional data and outputs the tag T.

Algorithm 3: Elephant v2 encryption algorithm enc

```
   Input: $(K, N, A, M) \in \{0,1\}^k \times \{0,1\}^m \times \{0,1\}^* \times \{0,1\}^*$
   Output: $(C, T) \in \{0,1\}^{|M|} \times \{0,1\}^t$
 1 $M_1, M_2, \ldots, M_{l_M} \overset{n}{\leftarrow} M$
 2 for $i \leftarrow 1$ to $l_M$ do
 3     $C_i \leftarrow M_i \oplus \mathsf{P}(N \| 0^{n-m} \oplus \mathsf{mask}_K^{i-1,1}) \oplus \mathsf{mask}_K^{i-1,1}$
 4 $C \leftarrow \lfloor C_1, \ldots, C_{l_M} \rfloor_{|M|}$
 5 $A_1, A_2, \ldots, A_{l_A} \overset{n}{\leftarrow} N \| A \| 1$
 6 $C_1, C_2, \ldots, C_{l_C} \overset{n}{\leftarrow} C \| 1$
 7 $T \leftarrow A_1$
 8 for $i \leftarrow 2$ to $l_A$ do
 9     $T \leftarrow T \oplus \mathsf{P}(A_i \oplus \mathsf{mask}_K^{i-1,0}) \oplus \mathsf{mask}_K^{i-1,0}$
10 for $i \leftarrow 1$ to $l_C$ do
11     $T \leftarrow T \oplus \mathsf{P}(C_i \oplus \mathsf{mask}_K^{i-1,2}) \oplus \mathsf{mask}_K^{i-1,2}$
12 $T \leftarrow \mathsf{P}(T \oplus \mathsf{mask}_K^{0,0}) \oplus \mathsf{mask}_K^{0,0}$
13 return $(C, \lfloor T \rfloor_t)$
```

Let $\mathsf{P} : \{0,1\}^n \to \{0,1\}^n$ be an n-bit permutation and $\varphi_1, \varphi_2 : \{0,1\}^n \to \{0,1\}^n$ be two masking LFSRs that satisfy $\varphi_2 = \varphi_1 \oplus \mathtt{id}$, where id is the identity function. The function $\mathsf{mask}_K^{a,b}$ is then defined as follows

$$\begin{aligned} \mathsf{mask}_K^{a,b} &: \{0,1\}^k \times \mathbb{N}^2 \to \{0,1\}^n \\ &(K, a, b) \mapsto \varphi_2^b \circ \varphi_1^a \circ \mathsf{P}\left(K \| 0^{n-k}\right), \end{aligned} \tag{5.1}$$

where the parameter $a, b \in \mathbb{N}$ for φ_1, φ_2 are used to ensure that every occurrence of the masking in the Elephant mode gets different parameters. For the encryption layer $(a, b) = (i, 1)$ is used, for the ciphertext authentication $(a, b) = (i, 2)$, and for associated data authentication $(a, b) = (i, 0)$.
Elephant consists of three instances with Dumbo being the primary member. The main difference is their parameterization, used permutation and masking LFSRs. The differences

are listed in the Table 5.1 below. The expected classical security strength is measured in terms of offline complexity, for example the amount of classical evaluations of the primitive done by the adversary.

Instance	k	m	n	t	P	expected classical security strength	Online complexity limit
Dumbo	128	96	160	64	Spongent-π[160][Bog+11]	2^{112}	2^{50} bytes
Jumbo	128	96	176	64	Spongent-π[176][Bog+11]	2^{127}	2^{50} bytes
Delirium	128	96	200	128	Keccak-f[200][Ber+11]	2^{127}	2^{74} bytes

Table 5.1: Parameterization of the Elephant instances.

The encryption process is illustrated in Figure 5.1. The ciphertext blocks can be generated in parallel since each block is independent from the others. The message is first padded as $M_1, \dots, M_{l_M} \overset{n}{\leftarrow} M$ then the ciphertext block C_i is generated as follows

$$C_i = M_i \oplus \mathtt{mask}_K^{i,2} \oplus \mathsf{P}\left(\mathtt{mask}_K^{i,2} \oplus (N \,\|\, 0^{n-m})\right). \tag{5.2}$$

The resulting ciphertext equals to $C = \lfloor C_1 \dots C_{l_M} \rfloor_{|M|}$.

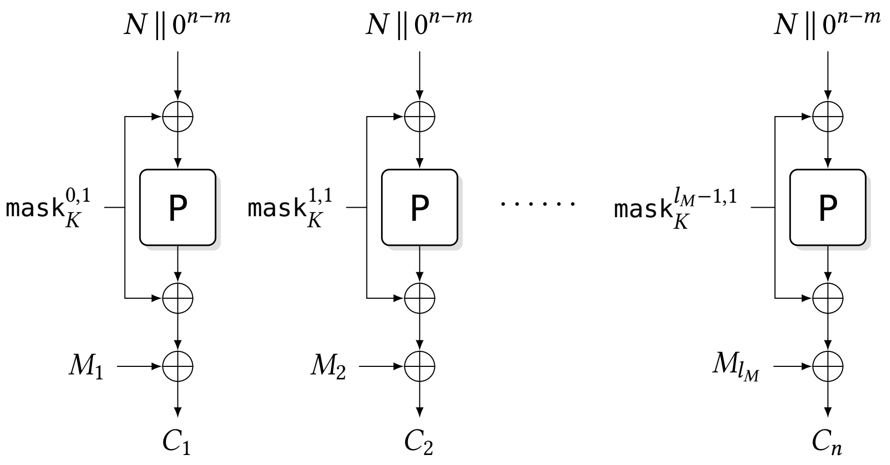


Figure 5.1: The encryption part of Elephant: Message is padded as $M_1, \dots, M_{l_M} \overset{n}{\leftarrow} M$ and the resulting ciphertext equals to $C = \lfloor C_1 \dots C_{l_M} \rfloor_{|M|}$. The rounded squares represent permutation calls.

The authentication mode of enc is illustrated in Figure 5.2. The nonce N is padded with the given associated data A and afterwards split into n-bit blocks

$$A_1, A_2, \dots, A_{l_A} \overset{n}{\leftarrow} A \,\|\, N \,\|\, 1. \tag{5.3}$$

The tag T is then generated with use of the padded associated data and the padded ciphertext $C_1, \dots C_{l_C} \overset{n}{\leftarrow} C \,\|\, 1$. The produced tag T is truncated to t bits and returned together with the ciphertext C.

This procedure is an encrypt-then-MAC mode, where the encryption is realized by counter mode and the message authentication by a variant of the counter sum [Luy+16; Ber99]. Both instantiated using the masked Even-Mansour procedure by Granger et al. [Gra+16a].

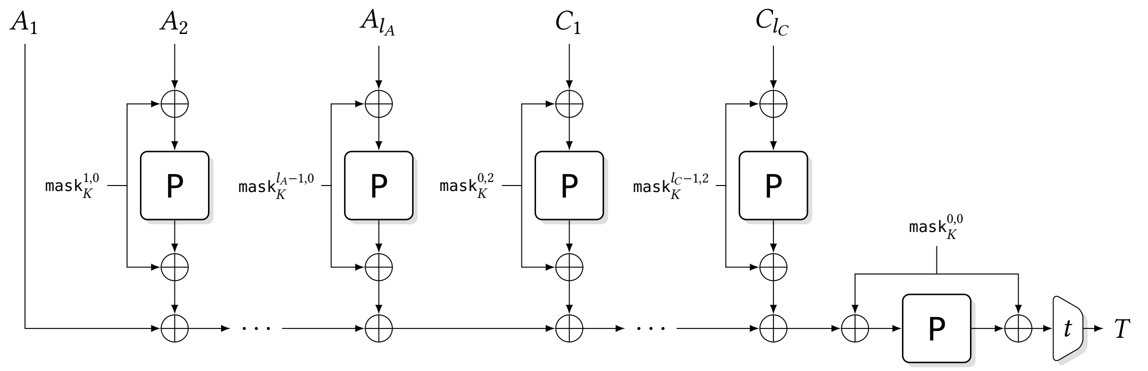


Figure 5.2: Authentication part: The nonce is prepended to the associated data and split into n-bit blocks $A_1, \dots, A_{l_A} \overset{n}{\leftarrow} N \,\|\, A \,\|\, 1$, the ciphertext is padded as $C_1, \dots, C_{l_C} \overset{n}{\leftarrow} C \,\|\, 1$.

In their security analysis [Men21, Section B] the creators of `Elephant` prove that in the ideal permutation model the mode of `Elephant` is structurally sound. They also emphasize that the `Elephant` security claims only hold in the nonce-respecting setting, the adversary may not evaluate the encryption function twice under the same nonce.

5.2.2 Quantum key recovery attack on Elephant v2

A brief summary of existing superposition attacks against the `Elephant` block cipher.

Version	Model	Queries	Type of Simon	Reference
v1.1	Q1	$O(2^{n/3})$	Offline Simon	Bonnetain and Jaques in [BJ20]
v1.1	Q2	$O((n+m)\cdot 2^{m/2})$	Grover-meets-Simon	this thesis in subsection 5.2.3 and the concurrent work by Shi et al. in [Shi+21]
v2	Q2	$O(n)$	Basic Simon	this thesis in subsection 5.2.2
v2	Q1	$O(2^{n/3})$	Offline Simon	this thesis in subsection 5.2.2

Table 5.2: Existing superposition attacks against the `Elephant` block cipher.

Remark: The attack presented by Shi et al. and the attack in subsection 5.2.3 are very similar but have been created independently from each other. The attack in subsection 5.2.3 takes a different approach when testing for the correct period. This reduces the number of queries required and makes the attack cheaper overall.

One of the changes from `Elephant` v1.1 to v2 was the change in the positioning in the masks. The round key $\mathtt{mask}_K^{0,0}$ is now used as the last step in the associated data authentication. As defined in Equation 5.1 the $\mathtt{mask}_K^{0,0}$ correspond to the expanded key $\mathsf{P}\left(K \,\|\, 0^{n-k}\right)$. Thus, an adversary having access to the expanded round key is able to recreate every other round key $\mathtt{mask}_K^{\cdot,\cdot}$ in `Elephant` , given that the LFSR φ_1, φ_2 are publicly known.

The attack on `Elephant` v2 is in the Q2 model and an adversary has access to an encryption Oracle in superposition. First the attack will be introduced without taking truncation into account. Looking at Figure 5.2 the adversary is only considering the final masking block with $\mathtt{mask}_K^{0,0}$. The empty message 0^n will be used as the message input, the nonce N and $(n-m-1)$-bits of arbitrary associated data x will be used to obtain $A_1 \leftarrow N \| x \| 1$ as seen in Equation 5.3 or line 5 in Algorithm 3.
Since the nonce N will be the input for this attack it has to be queried in superposition. Therefore the chance to reuse a nonce exists every time `enc` is called.
Out of convenience the associated data block A_1 will be defined as $N \| x$ without the final padded 1. Therefore the associated data x will be of length $(n-m)$ instead of $n-m-1$. This change has no impact on the attack. In addition $A_1 \in \{0,1\}^n$ will from now on be the input for the following functions. It is important to keep in mind that A_1 contains the nonce when applying the attack.
Querying `Elephant` with this input results in the following structure illustrated in Figure 5.3.

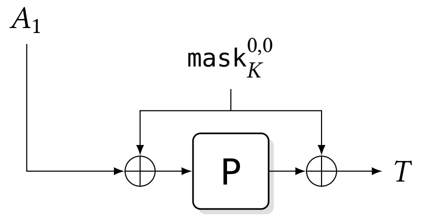


Figure 5.3: Tag creation of `Elephant` algorithm `enc` with input $A_1 \leftarrow N \| x$.

The function $\mathtt{tag}_1$ is then defined as a sub-algorithm of `enc` which only considers the tag-output T. The ciphertext can be ignored.

$$\begin{aligned} \mathtt{tag}_1 : \{0,1\}^n &\to \{0,1\}^n \\ A_1 &\mapsto \mathsf{P}\left(A_1 \oplus \mathtt{mask}_K^{0,0}\right) \oplus \mathtt{mask}_K^{0,0}, \end{aligned} \tag{5.4}$$

where $A_1 \leftarrow N \| x$ is defined as mentioned before. Thus the function f_1^{v2} can be defined as

$$\begin{aligned} f_1^{\mathrm{v2}} : \{0,1\}^n &\to \{0,1\}^n \\ A_1 &\mapsto \mathtt{tag}_1(A_1) \oplus \mathsf{P}(A_1) = \mathsf{P}\left(A_1 \oplus \mathtt{mask}_K^{0,0}\right) \oplus \mathtt{mask}_K^{0,0} \oplus \mathsf{P}(A_1). \end{aligned} \tag{5.5}$$

This function is periodic with $s = \mathtt{mask}_K^{0,0}$:

$$\begin{aligned} f_1^{\mathrm{v2}}(x) &= \mathsf{P}(A_1 \oplus \mathtt{mask}_K^{0,0}) \oplus \mathtt{mask}_K^{0,0} \oplus \mathsf{P}(A_1) \\ &= \mathsf{P}(A_1 \oplus \mathtt{mask}_K^{0,0}) \oplus \mathtt{mask}_K^{0,0} \oplus \mathsf{P}((A_1 \oplus \mathtt{mask}_K^{0,0}) \oplus \mathtt{mask}_K^{0,0}) \\ &= \mathsf{P}((A_1 \oplus \mathtt{mask}_K^{0,0}) \oplus \mathtt{mask}_K^{0,0}) \oplus \mathtt{mask}_K^{0,0} \oplus \mathsf{P}(A_1 \oplus \mathtt{mask}_K^{0,0}) \\ &= f_1^{\mathrm{v2}}(x \oplus \mathtt{mask}_K^{0,0}). \end{aligned} \tag{5.6}$$

Since the permutation is publicly known and the adversary has access to the encryption Oracle it is possible to create the unitary operator $U_{f_1^{\mathrm{v2}}}$. Using Simon's algorithm as seen in section 3.1 it is possible to extract $\texttt{mask}_K^{0,0}$ using only $O(n)$ calls to the encryption Oracle. Looking at the attack more closely shows that this is the Even-Mansour-construction, presented in subsection 3.5.1, using $k_0 = k_1 = \texttt{mask}_K^{0,0}$.

Dealing with truncation The just presented attack will be considered along with truncating the output to the t most significant bits. Therefore only t of $\texttt{mask}_K^{0,0}$ can be extracted using the following adaptation of the previous attack. First the function $\texttt{tag}_t$ is defined as

$$\begin{aligned} \texttt{tag}_t : \{0,1\}^n &\to \{0,1\}^t \\ A_1 &\mapsto \left\lfloor \mathsf{P}\left(A_1 \oplus \texttt{mask}_K^{0,0}\right) \oplus \texttt{mask}_K^{0,0} \right\rfloor_t . \end{aligned} \tag{5.7}$$

$\texttt{tag}_t$ can then be used to create the function f_t^{v2}.

$$\begin{aligned} f_t^{\mathrm{v2}} : \{0,1\}^n &\to \{0,1\}^t \\ A_1 &\mapsto \texttt{tag}_t(A_1) \oplus \lfloor \mathsf{P}(A_1) \rfloor_t = \left\lfloor \mathsf{P}\left(A_1 \oplus \texttt{mask}_K^{0,0}\right) \oplus \texttt{mask}_K^{0,0} \right\rfloor_t \oplus \lfloor \mathsf{P}(A_1) \rfloor_t . \end{aligned} \tag{5.8}$$

This function is still periodic with $s = \texttt{mask}_K^{0,0}$ as seen in Equation 5.6 and using the promise that $\forall x, y \in \{0,1\}^* : \lfloor x \oplus y \rfloor_t = \lfloor x \rfloor_t \oplus \lfloor y \rfloor_t$.
Using the results of section 3.4 it is still possible to apply Simon's algorithm to recover the period s regardless of the truncation to $t < n$ bits. Bonnetain proposed in [Bon20] that only $\lceil 3.5 + log_2(q) \rceil$ output bits are needed (see Heuristic 4) when making q queries and having a reversible implementation of Simon's algorithm.
Simon's algorithm will return $s = \texttt{mask}_K^{0,0}$ by running additional times to cancel out interferences caused by the occurring collisions. Let p_0 describe the expected probability of collisions to occur, since P is a considered to be a pseudorandom permutation $p_0 = \frac{2^{n-t}}{2^n}$. According to Theorem 1 in [Kap+16], Simon's algorithm will recover s within $O(cn)$ queries, where c can be chosen as $c \geq \frac{3}{(1-p_0)}$. Thus, for this attack choosing $c > 3$ will be enough to recover s.

Attack on Elephant v2 in Q1 As mentioned in the previous part, the presented attack is like the basic attack on Even-Mansour with $k_0 = k_1 = \texttt{mask}_K^{0,0}$. Therefore, the offline Simon's algorithm and the attack presented in subsection 3.5.4 can be applied to this version of Elephant .

The $\texttt{mask}_K^{0,0}$ will be divided into $\texttt{mask}_K^{0,0,(1)}$ of u-bits and $\texttt{mask}_K^{0,0,(2)}$ of $n-u$-bits. The $\texttt{mask}_K^{0,0,(2)}$ part will be guessed using Grover's algorithm and with the respective result of $\texttt{mask}_K^{0,0,(2)}$, it is possible to recover $\texttt{mask}_K^{0,0,(1)}$ using Simon's algorithm. The integer u will be set to $\frac{n}{3}$. The functions F and g used for the offline Simon's algorithm are defined by

$$\begin{aligned} F : \{0,1\}^{n-u} \times \{0,1\}^u &\to \{0,1\}^n, & g : \{0,1\}^u &\to \{0,1\}^n \\ i, x &\mapsto P(x \,\|\, i), & x &\mapsto \texttt{tag}_1(x \,\|\, 0^{n-u}). \end{aligned} \tag{5.9}$$

Using these two functions F and g to create the periodic attack function

$$\begin{aligned} f_o : \{0,1\}^{n-u} \times \{0,1\}^u &\to \{0,1\}^n \\ i, x &\mapsto F(i,x) \oplus g(x), \end{aligned} \tag{5.10}$$

that is periodic in $\texttt{mask}_K^{0,0,(1)}$ once the correct $i = \texttt{mask}_K^{0,0,(2)}$ was found. This can be shown as follows

$$\begin{aligned} f_o(\texttt{mask}_K^{0,0,(2)}, x) &= F(\texttt{mask}_K^{0,0,(2)}, x) \oplus g(x) \\ &= P(x \,\|\, \texttt{mask}_K^{0,0,(2)}) \oplus P((x \oplus \texttt{mask}_K^{0,0,(1)} \,\|\, \texttt{mask}_K^{0,0,(2)}) \oplus \texttt{mask}_K^{0,0} \\ &= f_o(\texttt{mask}_K^{0,0,(2)}, x \oplus \texttt{mask}_K^{0,0,(1)}). \end{aligned} \tag{5.11}$$

The period $s = \texttt{mask}_K^{0,0}$ can therefore be recovered using the offline Simon's algorithm described in section 3.3 together with the functions F, g and f_o and making $O(2^{n/3})$ queries. Since the output is getting truncated additional queries have to be made. As in the analysis for the Q2 version in the previous section concluded, choosing $c > 3$ is enough to recover s. The total cost of the attack is $O(c \cdot 2^{n/3})$. The attack is nonce respecting, since every query to the oracle will use a different value for x and therefore a different nonce N.

5.2.3 Quantum key recovery attack on Elephant v1.1

This attack on `Elephant v1.1` is in the Q2 model hence the encryption algorithm `enc` is accessible in superposition. Combining this fact, Leander and Mays approach on breaking the FX-construction [LM17] and adjusting the Equation 5.2 it is possible to perform an encryption key recovery attack on `Elephant` . Since this attack is independent from the used permutation it can be applied to every `Elephant` variation.

Changes from `Elephant` v2 There are only a few changes from `Elephant v1.1` to `Elephant` v2. The variant of Wegman-Carter-Shoup MAC function was replaced by a variant of the protected counter sum MAC function. The roles of the masks was changed as well. For associated data authentication the mask $(\cdot, 0)$, for encryption $(\cdot, 1)$ and for ciphertext authentication $(\cdot, 2)$ is now used.

Therefore the `Elephant v1.1` uses $(\cdot, 0)$ for encryption, $(\cdot, 1)$ for ciphertext authentication and $(\cdot, 2)$ for associated data authentication.

The changed algorithm of `Elephant v1.1` can be found below. Changes in the Algorithm besides different masks are in Line 7 where T is initialized with 0 instead of A_1, in Line 8 where the for-loop runs from 1 instead of 2, and there is no additional masked permutation block between Line 11 and returning the output. This attack targets the encryption algorithm of `Elephant v1.1`, besides the mask there was no change made to `Elephant` v2 see Figure 5.1.

Algorithm 4: Elephant v1.1 encryption algorithm enc

```
   Input: $(K, N, A, M) \in \{0,1\}^k \times \{0,1\}^m \times \{0,1\}^* \times \{0,1\}^*$
   Output: $(C, T) \in \{0,1\}^{|M|} \times \{0,1\}^t$
1  $M_1, M_2, \ldots, M_{l_M} \overset{n}{\leftarrow} M$
2  for $i \leftarrow 1$ to $l_M$ do
3      $C_i \leftarrow M_i \oplus \mathsf{P}(N \,\|\, 0^{n-m} \oplus \mathtt{mask}_K^{i-1,0}) \oplus \mathtt{mask}_K^{i-1,0}$
4  $C \leftarrow \lfloor C_1, \ldots, C_{l_M} \rfloor_{|M|}$
5  $A_1, A_2, \ldots, A_{l_A} \overset{n}{\leftarrow} N \,\|\, A \,\|\, 1$
6  $C_1, C_2, \ldots, C_{l_C} \overset{n}{\leftarrow} C \,\|\, 1$
7  $T \leftarrow 0$
8  for $i \leftarrow 1$ to $l_A$ do
9      $T \leftarrow T \oplus \mathsf{P}(A_i \oplus \mathtt{mask}_K^{i-1,2}) \oplus \mathtt{mask}_K^{i-1,2}$
10 for $i \leftarrow 1$ to $l_C$ do
11     $T \leftarrow T \oplus \mathsf{P}(C_i \oplus \mathtt{mask}_K^{i-1,1}) \oplus \mathtt{mask}_K^{i-1,1}$
12 return $(C, \lfloor T \rfloor_t)$
```

Only one encryption block of enc will be considered in the attack. The input in this case is the nonce N and not the message M_i. The message M_1 is not of importance and hence set to the empty message $M_1 = 0$. The fundamental idea is to transform Equation 5.2 into the following function. For convenience x will denote the nonce used as input and κ^i will denote the round key $\mathtt{mask}_K^{i,0}$. The superscript i will be dropped as well since only one encryption block will be considered.

$$\begin{aligned} f_1 : \{0,1\}^m &\to \{0,1\}^n \\ x &\mapsto C_1 \oplus \mathsf{P}(x \,\|\, 0^{n-m}) = \kappa \oplus \mathsf{P}(x \,\|\, 0^{n-m} \oplus \kappa) \oplus \mathsf{P}(x \,\|\, 0^{n-m}) \,. \end{aligned} \tag{5.12}$$

Even though the function seems to be periodic with $s = \kappa$ this is not the case. The last $(n - m)$ bits are unknown and therefore Simon's Algorithm is not applicable. The attack is possible if the last $(n - m)$ bits of the round key κ are known. This can be achieved by adjusting the function $f(x)$. Instead of XORing $\mathsf{P}(x \,\|\, 0^{n-m})$ with C_1 one can use $\mathsf{P}(x \,\|\, r)$ instead with arbitrary $r \in \{0,1\}^{n-m}$. This is similar to the attack of Leander and May in [LM17] if the last unknown $(n - m)$ bits of P are considered as a block cipher E with key r. This results in the following function

$$\begin{aligned} f_2 : \{0,1\}^m \times \{0,1\}^{n-m} &\to \{0,1\}^n \\ x, r &\mapsto C_1 \oplus \mathsf{P}(x \,\|\, r) = \kappa \oplus \mathsf{P}(x \,\|\, 0^{n-m} \oplus \kappa) \oplus \mathsf{P}(x \,\|\, r) \,. \end{aligned} \tag{5.13}$$

The round key can be written as $\kappa = \kappa_1 \,\|\, \kappa_2$ where κ_1 denote the first m-bits of κ and κ_2 the last $(n - m)$ bits. Once r is equal to κ_2 the function $f_2(x, r)$ is periodic with $s = \kappa_1$. The last steps in the following proof are inspired by [Shi+21] and [BJ20].

$$\begin{aligned} f_2(x,r) &= \kappa \oplus \mathsf{P}\left(x \,\|\, 0^{n-m} \oplus \kappa\right) \oplus \mathsf{P}\left(x \,\|\, r\right) \\ &= \kappa \oplus \mathsf{P}\left(x \,\|\, 0^{n-m} \oplus \kappa_1 \,\|\, \kappa_2\right) \oplus \mathsf{P}\left(x \,\|\, r\right) \\ &= \kappa \oplus \mathsf{P}\left((x \oplus \kappa_1) \,\|\, \kappa_2\right) \oplus \mathsf{P}\left(x \,\|\, \kappa_2\right) \\ &= \kappa \oplus \mathsf{P}\left((x \oplus \kappa_1) \,\|\, \kappa_2\right) \oplus \mathsf{P}\left((x \oplus \kappa_1 \oplus \kappa_1) \,\|\, \kappa_2\right) \\ &= \kappa \oplus \mathsf{P}\left((x \oplus \kappa_1 \oplus \kappa_1) \,\|\, \kappa_2\right) \oplus \mathsf{P}\left((x \oplus \kappa_1) \,\|\, \kappa_2\right) \\ &= \kappa \oplus \mathsf{P}\left((x \oplus \kappa_1) \,\|\, 0^{n-m} \oplus \kappa_1 \,\|\, \kappa_2\right) \oplus \mathsf{P}\left((x \oplus \kappa_1) \,\|\, \kappa_2\right) \\ &= \kappa \oplus \mathsf{P}\left((x \oplus \kappa_1) \,\|\, 0^{n-m} \oplus \kappa\right) \oplus \mathsf{P}\left((x \oplus \kappa_1) \,\|\, \kappa_2\right) \\ &= f_2(x \oplus \kappa_1, r). \end{aligned} \tag{5.14}$$

The Equation 5.14 shows that once the correct r is chosen Simon's algorithm can be applied and the round key κ extracted. The value of κ_2 can be found using the Grover-meets-Simon algorithm by Leander and May in [LM17].
The Grover-meets-Simon algorithm is presented in section 3.2. Let l denote the amount of computations run in parallel and as Leander and May suggest, choosing $l > 3n$ is sufficiently large enough. Let h be defined as the function evaluating $f_2(\cdot,\cdot)$ in parallel on l arguments in the first component. Thus,

$$\begin{aligned} h : (\{0,1\}^m)^l \times \{0,1\}^{n-m} &\to \{0,1\}^{n \cdot l} \\ (x_1, \dots, x_l, u) &\mapsto f(x_1,u) \,\|\, \dots \,\|\, f(x_l,u). \end{aligned} \tag{5.15}$$

Let the unitary operator U_h be the embedding of h on $k + 2nl$ qubits and map

$$|x_1, \dots, x_l, u, 0, \dots, 0\rangle \mapsto |x_1, \dots, x_l, u, h(x_1, \dots, x_l, u)\rangle .$$

Thus, the quantum algorithm $\mathcal{A}$ presented in section 3.2 can be applied. This results in a set of vectors $S = \{y_1, \dots, y_l\}$ orthogonal to the secret κ_2. Instead of testing if the l vectors of S span the $m-1$ dimensional subspace orthogonal to κ_2, one computes the basis s_l of S and checks if the function f fulfills $f_2(0,0) = f_2(u, s_l)$ [Bon20]. For the amplifying part of the Grover-meets-Simon algorithm on $\mathcal{A}$, the Classifier $\mathcal{B}$ is then defined as

$$\begin{aligned} \mathcal{B} : \{0,1\}^{(n-m)+ml} &\to \{0,1\} \\ (y_1, \dots, y_l, u) &\mapsto \begin{cases} \mathcal{B}\,|y_1, \dots, y_l, u\rangle = 1, & \text{if } f_2(0,0) = f_2(u, s_l), \\ \mathcal{B}\,|y_1, \dots, y_l, u\rangle = 0, & \text{if } f_2(0,0) \neq f_2(u, s_l). \end{cases} \end{aligned} \tag{5.16}$$

Let $S_{\mathcal{B}}$ be the operator that performs a phase shift of the good state

$$|x\rangle \mapsto \begin{cases} -|x\rangle, & \text{if } \mathcal{B}(x) = 1, \\ |x\rangle, & \text{if } \mathcal{B}(x) = 0. \end{cases} \tag{5.17}$$

Define the unitary operator $Q = -\mathcal{A} S_0 \mathcal{A}^{-1} S_{\mathcal{B}}$, as seen in section 3.2, where S_0 changes the sign of the amplitude only for the state $|0\rangle$. After computing $Q^\eta \mathcal{A}\,|0\rangle$, where $\eta \approx 2^{(n-m)/2}$, one is able to recover $s = \kappa_1 \,\|\, \kappa_2$ by making $O(m+n) \cdot 2^{m/2}$ oracle queries.
This attack is applicable to `Elephant` v2 since the encryption process did not change between v1.1 and v2, beside using $\mathtt{mask}_K^{0,2}$ instead of $\mathtt{mask}_K^{0,0}$. Looking at Equation 5.1, the only downside is that other round keys $\mathtt{mask}_K^{i,2}$ can not be generated using the extracted $\mathtt{mask}_K^{0,2}$.

5.3 LightMAC

`LightMAC` proposed by Luykx et al. in 2016 [Luy+16]. It uses a keyed block cipher as underlying construction and functions as a MAC mode of operation, a construction which aims to provide data authenticity for long messages. `LightMAC` can be used as a MAC and as a PRF as well. Contrary to other MAC modes of operation the provided security of `LightMAC` is independent from the message length. `LightMAC` can therefore be used for much longer messages.

5.3.1 Specification

`LightMAC` uses the block ciphers `PRESENT` [Bog+07] or `AES-128` [01]. `PRESENT` is a lightweight `64`-bit block cipher. The block cipher calls can be made in parallel and are based on Bernsteins's protected counter sum [Ber99]. When tagging messages an additional counter must be used. This counter uses up to half of the block size. Therefore one of `LightMAC` disadvantages is that its rate is low.

Algorithm 5: $\texttt{LightMAC}_{K_1,K_2}(M)$

```
   Input: $K_1, K_2 \in \{0,1\}^k, M \in \{0,1\}^{\leq 2^s(n-s)}$
   Output: $T \in \{0,1\}^t$
1  $V \leftarrow 0^n \in \{0,1\}^n$
2  $M_1, M_2, \ldots, M_{l_M} \xleftarrow{n-s} M$
3  for $i \leftarrow 1$ to $l_M - 1$ do
4      $V \leftarrow V \oplus E_{K_1}(i_s \| M_i)$
5  $V \leftarrow V \oplus (M_{l_M} \| 10^*)$
6  $T \leftarrow \lfloor E_{K_2}(V) \rfloor_t$
7  return $T$
```

Let $k, v, n, t \in \mathbb{N}$ with $t \leq n$ and $v \leq \frac{n}{2}$. The parameter t donates the length of the truncation to t bits and the designers of `LightMAC` advice to use full tag length $t = n$. The parameter v denotes the length of the counter. Both parameter v and t are publicly known and can not be For an integer $1 \leq i \leq 2^v$, i_v defines some v-bit constant with $1 \leq i \leq j \leq 2^v$ then $i_v \neq j_v$. Let $E_{K_{1/2}} : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a keyed block cipher with $K_1, K_2 \in \{0,1\}^k$, and a message M of length at most $2^{v(n-v)}$. The expected upper classical security bound for `LightMAC` is expected to be $(1 + \epsilon) \cdot \frac{q^2}{2^n}$, where $\epsilon \in O\left(\frac{1}{2^{n/2}-1}\right)$. When `LightMAC` is used as a PRF, Algorithm 5 fully describes how the output is produced.

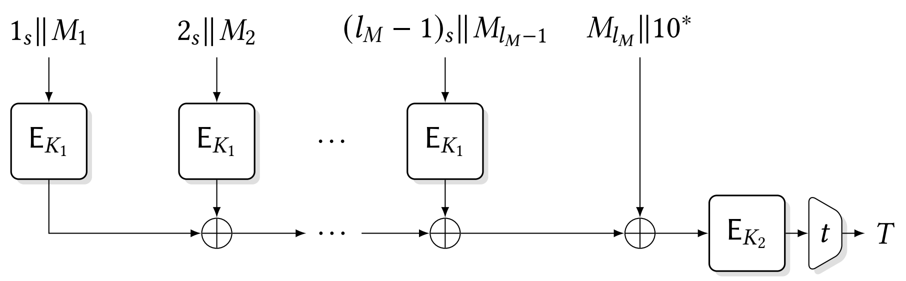


Figure 5.4: MAC mode of `LightMAC` : Message is padded as $M_1, \ldots, M_{l_M} \xleftarrow{n-v} M$. The rounded squares represent block cipher calls and the trapezium is a truncation to t bits.

The values for v can be chosen according to the needed purpose:

- $\boldsymbol{v = \frac{n}{2}}$: maximum supported message length (and hence with lowest rate $\frac{1}{2}$).
- $\boldsymbol{v = \frac{n}{3}}$: rounded to nearest multiple of 8 (with rate $\frac{2}{3}$).
- $\boldsymbol{v = 8}$: short maximum message length (with highest rate $(1 - \frac{8}{n})$).

To use `LightMAC` in the MAC or PRF mode, the message M is first split into l_M blocks of size $(n - v)$. The constants $i_v \in \{0,1\}^v$ are prepended to the first $l_M - 1$ message blocks. The last block $M_{l_M} \in \{0,1\}^{\leq n}$ is padded with 10^* to the length of n if needed. The MAC and PRF mode are illustrated in Figure 5.4. `LightMAC` gets two encryption keys $K_1, K_2 \in \{0,1\}^k$ for the underlying block cipher and the message $M \in \{0,1\}^{\leq 2^{v(n-v)}}$ as input. The parameter v and t must be agreed upon before a session starts, and remain constant during.

$$\begin{aligned} \texttt{LightMAC} : \{0,1\}^{2k} \times \{0,1\}^{\leq 2^{v(n-v)}} &\to \{0,1\}^t \\ (K_1, K_2, M) &\mapsto T = \left\lfloor E_{K_2}\left(M_{l_M} \,\|\, 10^* \oplus \bigoplus_{i=1}^{l_M - 1} E_{K_1}\left(i_v \,\|\, M_i\right)\right)\right\rfloor_t . \end{aligned} \tag{5.18}$$

When using the `LightMAC` as a PRF, the generated tag T is used as its output and when used as a MAC, the Tag T can be used to verify a message M as $T \overset{?}{=} \texttt{LightMAC}(M)$. The classical security analysis of the PRF and MAC mode can be found in [Luy+16, Section 4].

5.3.2 Distinction attack

A brief summary of existing superposition attacks against the `LightMAC` block cipher.

Model	Queries	Type of Simon	Reference
Q2	$O(n \cdot 2^{v+2})$	Basic	Haas in [Haa20]

Table 5.3: Existing superposition attacks against the `Elephant` block cipher

While the `LightMAC` PRF is classically proven to be indistinguishable from a uniformly distributed random function [Luy+16, Section 4.2] this assumption does not hold in a quantum setting. Jonas Haas showed in his masters thesis [Haa20, Section 4.2] that a distinction attack on `LightMAC` in a Q2 setting is possible given an oracle O that implements `LightMAC` . This attack is similar to the distinguishing attack on the three-round Feistel construction in subsection 3.5.2 and uses Simon's algorithm see section 3.1.

The attack will be introduced without taking truncation of the output into account. The authors of [Luy+16, Section 5.2] advise to use the full tag lengths $t = n$, hence 64−bit tags for `PRESENT` and 128−bit tags for `AES`. If instances still truncate the output, the results of section 3.4 are applicable. Therefore this attack still holds in a truncated settings.

Let $\alpha_0 \neq \alpha_1 \in \{0,1\}^{n-\nu}$ arbitrary but fixed, $x \in \{0,1\}^{n-\nu-1}$ and $b \in \{0,1\}$. The oracle O implements `LightMAC` as defined in Equation 5.18. Then the function f is defined as follows

$$\begin{aligned} f : \{0,1\} \times \{0,1\}^{n-\nu-1} &\longrightarrow \{0,1\}^n \\ b, x &\mapsto O(\alpha_b \,\|\, x) = E_{K_2}\left(x \,\|\, 10^\nu \oplus E_{K_1}(1_\nu \,\|\, \alpha_b)\right). \end{aligned} \tag{5.19}$$

The function f is illustrated in Figure 5.5 below.

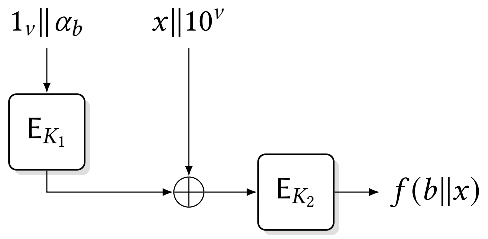


Figure 5.5: Function f used for the distinction attack with $\alpha_0 \neq \alpha_1 \in \{0,1\}^{n-\nu}$, $x \in \{0,1\}^{n-\nu-1}$ and $b \in \{0,1\}$.

Looking at f one can see that the function is periodic in $s = 1 \,\|\, E_{K_1}(1_\nu \,\|\, \alpha_0) \oplus E_{K_1}(1_\nu \,\|\, \alpha_1)$.

$$\begin{aligned} f(b \,\|\, x) = f(b' \,\|\, x') &\Leftrightarrow E_{K_1}(1_\nu \,\|\, \alpha_b) \oplus (x \,\|\, 10^\nu) = E_{K_1}(1_\nu \,\|\, \alpha_{b'}) \oplus (x' \,\|\, 10^\nu) \\ &\Leftrightarrow \begin{cases} x = x', & \text{if } b = b', \\ E_{K_1}(1_\nu \,\|\, \alpha_b) \oplus E_{K_1}(1_\nu \,\|\, \alpha_{b'}) = (x \,\|\, 10^\nu) \oplus (x' \,\|\, 10^\nu), & \text{if } b \neq b'. \end{cases} \end{aligned} \tag{5.20}$$

Haas remarked that the last $\nu + 1$ bits of the term $E_{K_1}(1_\nu \,\|\, \alpha_0) \oplus E_{K_1}(1_\nu \,\|\, \alpha_1)$ have to be 0 for this attack to work. The probability of this to happening is about $\frac{1}{2^{\nu+1}}$, since E is considered to be a pseudorandom function. This would result in an overall cost of $\mathcal{O}(2^{\nu+2} \cdot n)$ queries to the oracle.

However, using the result of section 3.4 shows that the attack is still applicable even if the last $\nu + 1$ bits are arbitrary or unknown. Therefore, the attack works every time and the cost can be reduced from $\mathcal{O}(2^{\nu+2} \cdot n)$ to $\mathcal{O}(2n)$ oracle queries.
Let U_f be the unitary operator that implements f using the oracle O. Afterwards perform Simon's algorithm subroutine until $n - 1$ linear independent vectors are found or at most $2n$ times. If not enough linear independent vectors were found, assume that O implements a random function and not `LightMAC` .

If enough independent vectors were found, it is possible to reconstruct s and query the oracle for arbitrary $x \in \{0,1\}^n$ and $x \oplus s$. If $O(x) = O(x \oplus s)$ holds, the oracle O implements LightMAC , if not the oracle implements a random function.
An adversary is therefore able to distinguish between LightMAC and a random function within $\mathcal{O}(n)$ time and making at most $\mathcal{O}(2n)$ queries to the oracle O.

5.4 ESTATE

ESTATE was first proposed by Chakraborti et al. in 2019 [Cha+20]. It is an energy efficient Single-state Tweakable block cipher based MAC-Then-Encrypt authenticated encryption scheme. ESTATE was proposed as a candidate to the NIST lightweight standardization process. It made it to the second round but was not chosen as a final candidate afterwards. To instantiate ESTATE two tweakable block ciphers TweGIFT-128 and TweAES-128 were proposed by the authors. TweAES-128 is a tweakable variant of AES-128 [01] block cipher and TweGIFT-128 the tweakable version of the GIFT-128 [Ban+17] block cipher. Both tweakable block ciphers are designed for efficient processing of small tweaks of size 4 bits.

5.4.1 Specification

The ESTATE mode of operation using the block ciphers TweAES-128 and TweGIFT-128 for instantiating will be presented in this section.
Let $n, \tau, k, t \in \mathbb{N}$ denote the *block size*, *tweak size*, *key size*, and *tag size*. The authors of ESTATE advice to fix those values to $n = 128$, $\tau = 4$, $k = 128$, and $t = n$. Throughout this section these values will be set as shown and otherwise will be noted.
The ESTATE authentication mode receives a key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^n$, an associated data $A \in \{0,1\}^*$, and a message $M \in \{0,1\}^*$ as inputs. It returns a tag $T \in \{0,1\}^t$ and ciphertext $C \in \{0,1\}^{|M|}$. The description of the decryption algorithm is not important for this attack but is specified in [Cha+20].
ESTATE is composed of an FCBC [BR05] like MAC, called *FCBC**, and the OFB [Dwo+16] mode of encryption, where the tweak value is fixed to 0. It operates on $n-$bit data blocks at a time using the tweakable block cipher $E_K^{\mathrm{T}}(M)$ with inputs $K \in \{0,1\}^k$, $\mathrm{T} \in \{0,1\}^\tau$, $M \in \{0,1\}^n$. Algorithm 6 presents the specification of ESTATE and the illustrated description is given in Figure 5.6, Figure 5.7, and Figure 5.8.
The functions *ozp*, *chop* and the conditional ?-operator are used in the following description of ESTATE . Since they may not be familiar, they will be briefly be defined below. The one-zero-padding function *ozp* maps bit-strings with smaller length than n to blocks of n-bit length by

$$ozp(x) : \{0,1\}^{\leq n} \longrightarrow \{0,1\}^n$$
$$x \longmapsto \begin{cases} 0^{n-|x|-1} \,\|\, 1 \,\|\, x, & \text{if } |x| < n, \\ x, & \text{else.} \end{cases}$$

The function *chop* takes a string x and an integer $i \leq |x|$ and returns the least significant i-bits of x. The conditional ? :::-operator functions on some predicates E_1 and E_2 with possible evaluations a, b, c and d. Then the ? ::: is defined as follows

$$(E_1; E_2)?a : b : c : d := \begin{cases} a, & \text{if } E_1 \wedge E_2, \\ b, & \text{if } E_1 \wedge \neg E_2, \\ c, & \text{if } \neg E_1 \wedge E_2, \\ d, & \text{if } \neg E_1 \wedge \neg E_2. \end{cases}$$

Algorithm 6: ESTATE Authenticated Encryption Algorithm

```
Function ESTATE.Enc[E](K,N,A,M)
    $T \leftarrow MAC(K, N, A, M)$
    $C \leftarrow OFB(K, T, M)$
    return $(C, T)$
Function ESTATE.MAC[E](K,N,A,M)
    if $|A| = 0$ and $|M| = 0$ then
        return $T \leftarrow E_K^8(N)$
    $T \leftarrow E_K^1(N)$
    if $|A| > 0$ then
        $A_{a-1} \| \ldots \| A_0 \leftarrow A$
        $t \leftarrow (|M| > 0; |A_{a-1}| = n)\ ?\ 2 : 3 : 6 : 7$
        $T \leftarrow FCBC^*[E](K, T, A, t)$
    if $|M| > 0$ then
        $M_{m-1} \| \ldots \| M_0 \leftarrow M$
        $t \leftarrow (|M_{m-1}| = n)\ ?\ 4 : 5$
        $T \leftarrow FCBC^*[E](K, T, M, t)$
    return $T$
Function FCBC*[E](K,T,D,t)
    $D_{d-1} \| \ldots \| D_0 \leftarrow D$
    for $i = 0$ to $d - 2$ do
        $T \leftarrow E_K^0(T \oplus D_i)$
    $T \leftarrow E_K^1(T \oplus ozp(D_{d-1}))$
    return $T$
Function OFB[E](K,T,M)
    $M_{m-1} \| \ldots \| M_0 \leftarrow M$
    for $i = 0$ to $m - 1$ do
        $T \leftarrow E_K^0(T)$
        $C_i \leftarrow chop(T, |M_i|) \oplus M_i$
    return $C_{m-1} \| \ldots \| C_0$
```

The tag generation phase using $FCBC^*$ is a tweakable version of FCBC, where the different twinks are used to instantiate the block cipher. The distinctness in the tweaks is used to distinguish between different cases based on the length of associated data and message. The 4-bit binary representation of integer i denotes the tweak value i. The first block, the nonce N, uses the tweak value 1. The following blocks are always processed with tweak 0. The last block may use tweak 2,4,6 if the block is full or 3,5,7 if the block is partial. The tag generation and the following OFB based encryption process with non-empty associated data and message, is illustrated in Figure 5.6. With neither the associated data nor the message being empty, the tag and ciphertext generation is given by

$$\begin{aligned} T &= E_K^{4/5}(M_{m-1} \oplus E_K^0(\dots(M_0 \oplus E_K^{2/3}(A_{a-1} \oplus E_K^0(\dots(A_0 \oplus E_K^1(N))\dots))\dots)) \\ C_i &= (E_K^0)^{i+1}(T) \oplus M_i. \end{aligned} \tag{5.21}$$

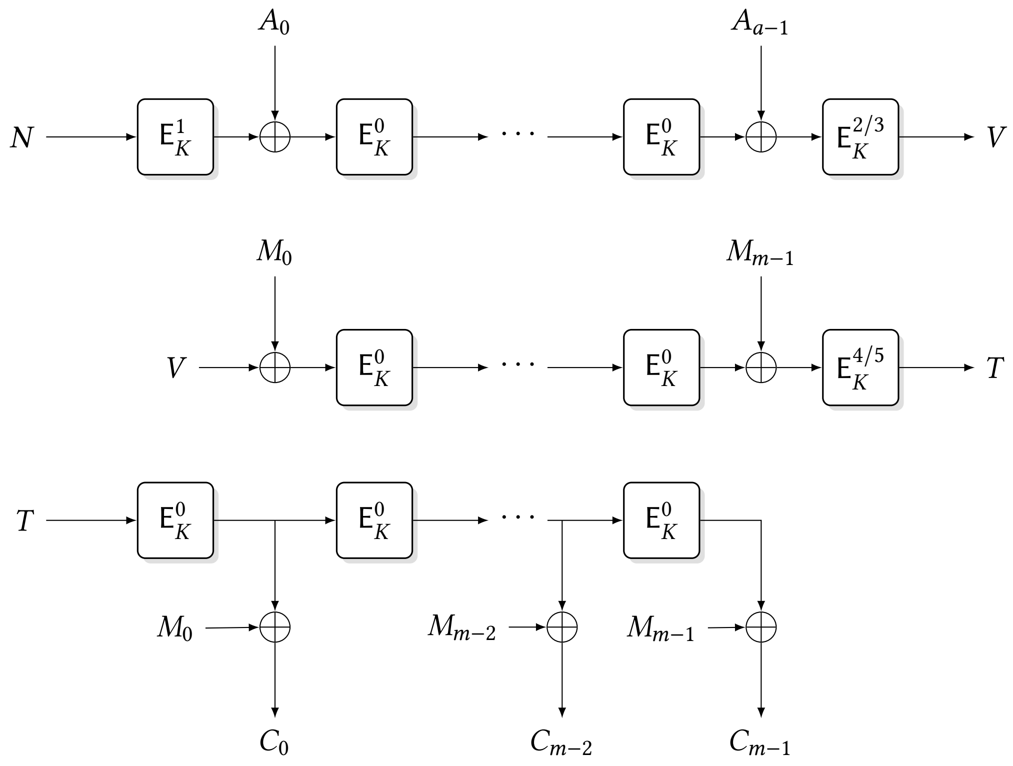


Figure 5.6: ESTATE with a associated data blocks and m message blocks.

ESTATE allows to query an instance with empty associated data, empty message or both. The empty associated data will be denoted as λ and the empty message as ϵ. The case where the associated data is empty is illustrated in Figure 5.7 and in this case the tag generation is given by

$$T = E_K^{4/5}(M_{m-1} \oplus E_K^0(\dots(M_0 \oplus E_K^1(N))\dots)). \tag{5.22}$$

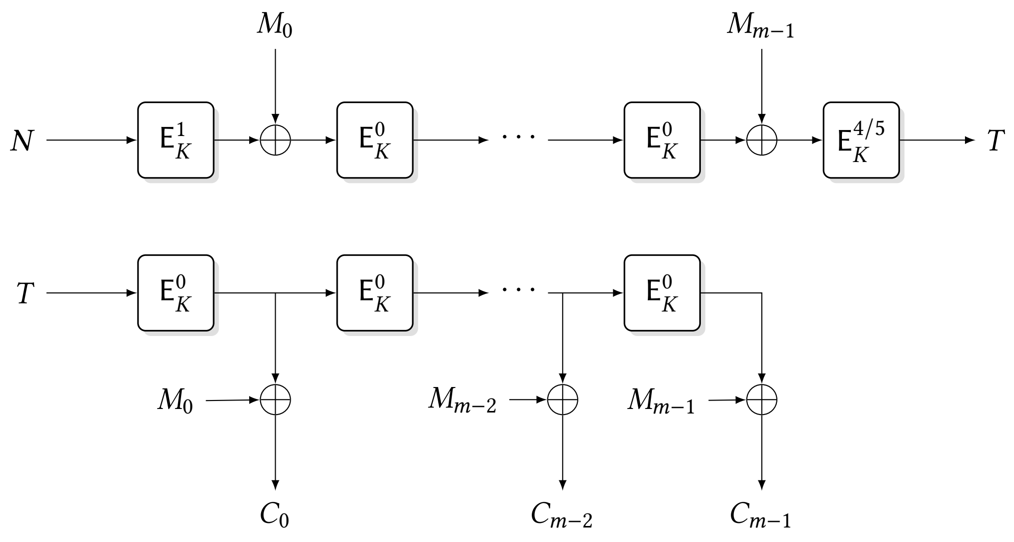


Figure 5.7: ESTATE with empty associated data and m message blocks.

At last the tag generation with empty message is illustrated in Figure 5.8 and given by

$$T = E_K^{6/7}(A_{a-1} \oplus E_K^0(\ldots(A_0 \oplus E_K^1(N))\ldots)) \tag{5.23}$$

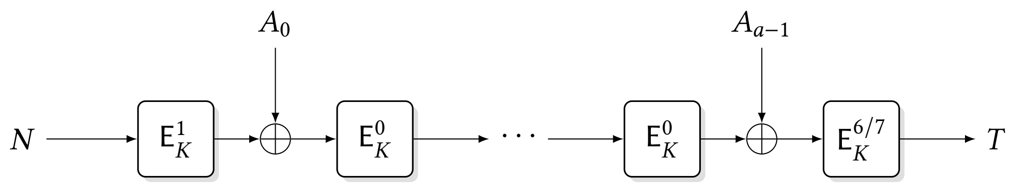


Figure 5.8: ESTATE with a associated data blocks and empty message λ.

5.4.2 Tag forgery attack

A brief summary of existing superposition attacks against the ESTATE block cipher.

Model	Queries	Type of Simon	Type of Attack	Reference
Q2	$O(n)$	Basic	Tag forgery attack	Haas in [Haa20]
Q2 (sEstate)	$O(2^{n/3})$	Simon and quantum square attack	Key recovery attack	Shi et al. in [Shi+21]

Table 5.4: Existing superposition attacks against the Elephant block cipher

This tag forgery attack was presented by Jonas Haas in his masters thesis [Haa20]. The attack is set in the Q2 model and in contrary to the other attack this adversary is not nonce respecting. The adversary is able to forge tags to almost arbitrary messages, authenticated data or ciphertexts. The only restriction being that all but one block can be chosen freely. The nonce N will be fixed and therefore this attack is not nonce respecting since the same nonce will be reused.

Let $\beta = \beta_0, \beta_1, \dots, \beta_{l_\beta-2}.\alpha_1$ denote the $l_\beta \cdot n$-bit prefix of the message for which a tag is forged. For $\alpha_0 \neq \alpha_1$, let β' denote the prefix $\beta_0, \beta_1, \dots, \beta_{l_\beta-2}.\alpha_0$ and $\gamma = \gamma_0, \dots, \gamma_{l_\gamma-1}$ the $l_\gamma \cdot n$-bit suffix of the message.
The function $tag(a, m)$ models a call to the oracle O implementing the tag generation part of `ESTATE` , hence the `ESTATE.MAC`$[E](K, N, A, M)$ part of Algorithm 6.
For parameters $b \in \{0,1\}$ and $x \in \{0,1\}^n$, consider the function f

$$\begin{aligned} f : \{0,1\}^{n+1} &\to \{0,1\}^n \\ b, x &\mapsto tag(\lambda, \beta_0 \,\|\, \beta_1 \,\|\, \dots \beta_{l_\beta-2} \,\|\, \alpha_b \,\|\, x) \\ &= E_K^4(x \oplus E_K^0(\alpha_b \oplus \underbrace{E_K^0(\beta_{l_\beta-2} \oplus \dots E_K^0(\beta_0 \oplus E_K^1(N)) \dots)}_{:=\phi})). \end{aligned} \tag{5.24}$$

The unitary operator U_f that implements the function f can be constructed using a single call to the oracle O. The function f is periodic within $s = 1 \,\|\, E_K^0(\alpha_0 \oplus \phi) \oplus E_K^0(\alpha_1 \oplus \phi)$ since

$$\begin{aligned} f(b \,\|\, x) = f(b' \,\|\, x') &\Leftrightarrow E_K^4(x \oplus E_K^0(\alpha_b \oplus \phi)) = E_K^4(x' \oplus E_K^0(\alpha_{b'} \oplus \phi)) \\ &\Leftrightarrow x \oplus E_K^0(\alpha_b \oplus \phi) = x' \oplus E_K^0(\alpha_{b'} \oplus \phi) \\ &\Leftrightarrow \begin{cases} x = x', & \text{if } b = b', \\ E_K^0(\alpha_0 \oplus \phi) \oplus E_K^0(\alpha_1 \oplus \phi) = x \oplus x', & \text{if } b \neq b'. \end{cases} \end{aligned} \tag{5.25}$$

Let s' denote the period s without the leading bit 1. Then an adversary is able to forge a tag in the Q2-EUF-CMA experiment by

1. Reconstruct s using Simon's algorithm.
2. Perform a classical query with empty associated data and the message $M' = \beta_0, \beta_1, \dots, \beta_{l_\beta-2}, \alpha_0, 0^n, \gamma_0, \gamma_1, \dots, \gamma_{l_\gamma-1}$ to retrieve the corresponding tag T and ciphertext $C' = C_0, C_1, \dots, C_{l_\beta+l_\gamma}$.
3. Output the tuple $((\lambda, M), T, C)$ as the forgery in the experiment, with $M = \beta_0, \beta_1, \dots, \beta_{l_\beta-2}, \alpha_0, s', \gamma_0, \gamma_1, \dots, \gamma_{l_\gamma-1}$, the same tag T as in the previous query and the ciphertext $C = C_0, \dots, C_{l_\beta-2}, C_{l_\beta-1} \oplus \alpha_0 \oplus \alpha_1, C_{l_\beta} \oplus s', C_{l_\beta+1}, \dots, C_{l_\beta+l_\gamma}$.

The tag is valid for both messages M and M' since the state after applying the l_β-th encryption step results in

$$E_K^0(\alpha_1 \oplus \phi) \oplus (E_K^0(\alpha_0 \oplus \phi) \oplus E_K^0(\alpha_1 \oplus \phi)) = E_K^0(\alpha_0 \oplus \phi). \tag{5.26}$$

and hence the same state as generating the tag for M'. The ciphertext is correct since the tag is valid for each of the messages M and M'.
The adversary is able to forge the tag and win the EUF-CMA experiment by making at most $2n + 2$ queries to the oracle O. Therefore `ESTATE` is not able to hold the Q2-EUF-CMA security for adversaries that are not nonce respecting. This attack can not be applied if the settings requires the adversary to be nonce respecting, like the attack model presented by Kaplan et al. in [Kap+16], where the nonce is randomly chosen by the oracle.
Haas notes in [Haa20] that the attack is also adjustable to work with arbitrary fixed associated data and empty message.

6 Cost analyses

In this chapter the resource cost of the superposition attacks presented in chapter 5 will be estimated. They will be analyzed considering the application, logical, and quantum error correction layer as introduced in section 2.3. The fault tolerant quantum computer to perform these attacks is based on the surface code. All attacks are following the approaches by Amy et al. in [Amy+17], Berger and Tiepelt in [BT21] and Fowler et al. in [Fow+12]. The goal of this chapter is to provide comparable results of the different attacks presented in this thesis.

6.1 Introduction and overview

For all attacks the permutations and block ciphers suggested by the authors will be used in the following cost analyses. Thus, SPONGENT and KECCAK for the corresponding instances of `Elephant` , GIFT128 and AES-128 for ESTATE and AES-128 for `LightMAC` . There are different resource analyzes of the quantum circuits for AES-128 and GIFT, like for example the AES implementation by Grassl et al. presented 2016 in [Gra+16b]. Zou et al. improved Grassl et al. work in [Zou+20] by presenting quantum circuit implementations of AES with fewer qubits. In their work they proposed a circuit using fewer qubits for the S-box of the AES. While the circuit presented by Grassl et al. needs a total of 984 qubits, the one presented by Zou et al. only needs 512 qubits in total. Whereas these works focus on minimizing the required amount of qubits for their circuits, hence the circuit's width, Samuel Jaques et al. have taken a different approach in [Jaq+20] on improving the circuit. They optimized the circuit under a depth restriction and introduced techniques that reduce the depth even if that requires more qubits. Since this approach also lowers the T-depth and needed T-gates, the overall cost for an attack using this AES circuit is lower. Therefore, the results by Jaques et al. will be used for the following cost analyses using the AES block cipher. For the GIFT block cipher, the results of Jang et al. in [Jan+21] and Bijwe et al. [BCS20] were considered. Like the reasoning given above, the results by Bijwe et al. will be used for the attacks using the GIFT block cipher. For the `Elephant` instances, the results by Bonnetain in [BJ20] were used.

6.2 Resource estimate

The resource estimations in this chapter are following the approaches by Amy et al. in [Amy+17], Berger and Tiepelt in [BT21] and Fowler et al. in [Fow+12].

Assumption 1. *[Assumption 1 in [Amy+17]] The resources needed for a large fault-tolerant quantum computation are well approximated using a surface code based quantum computer.*

The parameters below consider a fault-tolerant quantum computer using surface codes and approximately correspond to the current state of the art.

Assumption 2. *[[Amy+17; Fow+12; Jon+12]] Let $p_{in} \in \{10^{-2}, 10^{-3}, 10^{-4}\}$ define the initial injection error rate of a given quantum state, the error rate for gates is then $p_g \approx p_{in}/10$. Fowler et al. suggest that $t_{sc} = 200ns$ is a reasonable time for a single surface code cycle.*

Assumption 3. *All quantum gates are uniformly distributed over all layers.*

Assumption 4. *(Cost metric [Amy+17]) Let σ define the surface code cycles needed for a quantum computation implementing l logical qubits. The cost of the quantum computation is then equal to the classical cost of calling a block cipher $l \cdot \sigma$ times.*

When considering the needed resources to implement gates into surface codes, the fault tolerant T-gate is the most expensive one. A high fidelity logical implementation of a T-gate uses special ancilla states called magic states. Each T-gate relies on the

$$|A_L\rangle = \frac{1}{\sqrt{2}}\left(|0_L\rangle + e^{i\pi/4}|1_L\rangle\right) \tag{6.1}$$

magic state, which is created by a special qubit in an arbitrary but imperfect state. This process is called state injection, with an initial error rate of p_{in}. The imperfect state is then purified by repeating a probabilistic process called *distillation* until the desired fidelity of the magic state is achieved. For the entire attack circuit to successfully run, the error rate of the purified $|A_L\rangle$ magic states has to be below $p_{out} = 1/T_c^g$. The distillation process and the so called distilleries, requires a large amount of logical and physical qubits.

Quantum error correction layer Let d define the distance of the implemented surface code, p_{in} the initial injection error rate of a given quantum state and p_{th} the threshold error per injection state of the surface code. The constants C_1, C_2 are determined by the specific surface code implementation. The maximum error per logical gate p_{out} can then be approximated as

$$p_{out} \approx C_1\left(C_2\frac{p_{in}}{p_{th}}\right)^{\left\lfloor \frac{d+1}{2}\right\rfloor}, \tag{6.2}$$

as seen in [Jon+12, Section IV A.]. For the number of physical qubits per logical qubit, the results of Gidney and Ekera in [GE21] are used, which, for a surface code of distance d, state that each logical qubit requires $2 \cdot (d+1)^2$ physical qubits.

Logic layer For distilling the $|A_L\rangle$ ancilla states, the 15-qubit Reed-Muller encoding presented in [Fow+12; RHG06; BK05b], is used. In this scheme, a logical Bell pair is created and one of the logical qubits from the pair is entangled with 14 other logical qubits. The 15 qubits are then rotated with the help of $T_L^\dagger$-gates, where each gate uses an approximation of the $|A_L\rangle$ state as an ancilla. These ancillas are either obtained by a state injection or from the previous round of distillation. From the error probability of an initial injected state is p_{in}, it follows that the error rate of the output state p_{dist} will be approximately $35p_{in}^3$. Logical errors that may appear during the distillation process will be ignored, following the work of [Amy+17; FDJ13].

The term $\epsilon \cdot p_{dist}$ describes the introduced amount of logical errors and hence $p_{out} = (1+\epsilon)p_{dist}$ and thus, $p_{in} \approx \sqrt[3]{p_{out}/35 \cdot p_{out}}$. The case in which as many new errors are generated as are eliminated by distillation is called the balanced case with $\epsilon = 1$. The physical error rate per gate in the surface code is about ten times smaller than p_{in}, hence $p_g = p_{in}/10$. The distillation process is performed over multiple rounds until p_out falls below the required error rate. An algorithm that determines the number of rounds, or layers, i and the corresponding distances d_i of the surface codes is given in Algorithm 7 by Amy et al. in [Amy+17].

d_1 denotes the distance of the surface code used for the initial Reed-Muller circuit, hence the top layer of distillation. Each round of distillation takes $10 \cdot d_i$ surface code cycles. The number of required logical qubits per layer equals to $Q_i^{log} = 16 \cdot 15^{i-1}$ and the required physical qubits depend on the respective distance d_i of the layer.

Application Layer In the implementation of the attack circuits, the number of logical $T-$, CNOT and single qubit gates as well as the T-depth and width is considered. Let T_U define be the number of logical T-gates and T_U^d the T-depth. The logical T-gates that can be done in parallel for each layer of depth is denoted as the T-width of the circuit and is computed as $T_U^w = T_U/T_U^d$. The total number of T-gates needed to implement Simon's algorithm, using the circuit U, is denoted by $T_{SA,U}$. Thus, $T_{SI,U}$ is the number of T-gates required for an iteration of Simon's algorithm. This notation also applies for the depth and width.
Let σ_{dist} define the total number of surface codes cycles needed to perform the distillation procedure and λ_{SA} the required number of iteration for Simon's algorithm. The total number of surface code cycles is therefore

$$\sigma_{SA} = \lambda_{SA} \cdot \sigma_{dist} \cdot T_U^d. \tag{6.3}$$

Let Q_U^{log} define the needed logical qubits to implement the circuit U and Q_{dist}^{log} the total number of logical qubits required for the magic state distillation process. The number of logic-qubit-cycles as defined in Assumption 4 and [Amy+17; BT21] is then considered to be the total cost of the attack

$$cost_U = \left(Q_U^{log} + Q_{dist}^{log}\right) \cdot \sigma_{SA}. \tag{6.4}$$

6.3 Overview of a cost analysis

The attacks presented in chapter 5 will be analyzed. All different instances of the corresponding cipher will be considered, as well as the suggested implementations proposed by the authors. Every cost analysis will be carried out by examining the initial injection error tolerance of a given quantum state $p_{in} = \{10^{-4}, 10^{-3}, 10^{-2}\}$, as this is the range that is reasonably achievable in the current state of art [Fow+12]. For the magic state distillation only $\epsilon = 1$, hence the balanced case, will be considered in the following cost estimates. This is the case in which as many new errors are generated as are eliminated by distillation.

Overview The following section describes the process of a cost analysis. The respective results of the individual attacks are presented in section 6.4, section 6.5 and section 6.6.

Let U define the scheme being analyzed. Using the notation of the previous section, let T_U define the number of logical T-gates needed in the circuit implementing U, $T_{SI,U}$ the number of T-gates in one iteration of Simon's algorithm and λ_{SA} the required number of iteration for Simon's algorithm. The logical T-gates needed for Simon's algorithm is then

$$T_{SA} = \lambda_{SA} \cdot T^c_{SI}. \tag{6.5}$$

The exact value of T_{SI} depends on the used variant of Simon's algorithm and is explained in more detail in the respective sections. With the just obtained result for the required T-gates, the target output error rate for the state distillation can be determined as $p_{out} = 1/T_{SA}$. The needed layers i and distances d_i of the surface codes for p_{out} can be computed using Algorithm 7 by Amy et al. in [Amy+17, Algorithm 4].

Algorithm 7: Determining the required number of rounds of magic state distillation and the distances of the concatenated codes [Amy+17, Algorithm 4]

```
   Input: $\epsilon, p_q, p_g, p_{out}$
   Output: $d = [d_1, \dots, d_i]$
1  $d \leftarrow$ empty List []
2  $p \leftarrow p_{out}$
3  $i \leftarrow 0$
4  repeat
5      $i \leftarrow i + 1$
6      $p_i \leftarrow p$
7      Find minimum $d_i$ such that $192 d_i (100 p_g)^{\frac{d_i+1}{2}} < \frac{\epsilon p_i}{1+\epsilon}$
8      $p \leftarrow \sqrt[3]{p_i/(35(1+\epsilon))}$
9      $d.append(d_i)$
10 until $p > p_{in}$
```

The required number of physical qubits per logical qubit is $2 \cdot (d+1)^2$. The total logical qubits and physical qubits per layer i can then be computed as

$$\begin{aligned} Q_i^{log} &= 16 \cdot 15^{i-1}. \\ Q_i^{phy} &= Q_i^{log} \cdot 2 \cdot (d+1)^2 = 16 \cdot 15^{i-1} \cdot 2 \cdot (d+1)^2. \end{aligned} \tag{6.6}$$

The most bottom layer of the circuit occupies therefore the most physical qubits in the surface code. The physical qubits can be reused in the higher layers. Therefore, for a single distillery, the total required number of physical and logical qubits to get a purified $|A_L\rangle$ state is that of the most bottom layer. They will be noted as Q^{phy}_{dist} and Q^{log}_{dist}.
The needed surface code cycles per layer of distillation equals to $\sigma^i_{dist} 10 \cdot d_i$. The total number of cycles for the magic state distillation is thus

$$\sigma_{dist} = \sum_{j=0}^{i} \sigma^j_{dist}. \tag{6.7}$$

Reusing qubits for the middle layers allows potentially pipelining the distillation process to produce multiple magic states in parallel. The ratio

$$\phi = Q^{phy}_{dist} / Q^{phy}_{middle} \tag{6.8}$$

describes how many $|A_L\rangle$ can be produced in parallel in one distillation round σ_{dist}. The ϕ many magic states can therefore be produced every $\sigma_{dist} \cdot t_{sc}$. The required T_{SA} many states can be produced by a single magic state distillery in $t_{single} = T_{SA}/\phi \cdot \sigma_{dist} \cdot t_{sc}$ time.

To determine the needed distance for embedding the algorithms into a single surface code, the amount of Clifford gates in a single iteration of Simon's algorithm has to be taken into account. In the considered cases, the number of Clifford gates consists of CNOT and single qubit gates. They will be refereed to as Cl^{CNOT}_{U}, Cl^{1QC}_{U} and accordingly $CL^{CNOT/1QC}_{SA}$ and $CL^{CNOT/1QC}_{SI}$. The total amount of Clifford gates is then

$$Cl_{SA} \approx \lambda_{SA} \cdot (Cl^{CNOT}_{U} + Cl^{1QC}_{U}). \tag{6.9}$$

The error rate of the circuit should therefore be lower than

$$p^{A}_{out} = 1/Cl_{SA}. \tag{6.10}$$

For this, the smallest d_U is required that satisfies the following inequality [FDJ13]

$$\left(\frac{p_{in}}{0.0125}\right)^{\frac{d_U+1}{2}} < p^{A}_{out}. \tag{6.11}$$

Here the number of logical qubits depends solely on the implemented circuit U. However, the number of required physical qubits depends on the distance d_U of the surface code

$$Q^{phy}_{SA} = Q^{log}_{U} \cdot 2 \cdot (d_U + 1)^2. \tag{6.12}$$

Furthermore, the logical T-gates that can be done in parallel for each layer of depth are denoted as the T-width of the circuit and are computed as $T^{w}_{SI} = T_{SI}/T^{d}_{SI}$. To keep up computing the T^{w}_{SI} T-gates in the same σ_{dist} amount of cycles multiple magic state distilleries might be needed. The approximated number can be computed knowing that ϕ many states can be distilled in σ_{dist} cycles, hence one needs

$$\Phi = \lceil T^{w}_{SI}/\phi \rceil \tag{6.13}$$

many magic state factories working in parallel. This increases the needed physical qubits for the distillation part to

$$Q^{phy}_{total} = \Phi \cdot Q^{phy}_{dist}. \tag{6.14}$$

With Assumption 3 the average number of Clifford gates per layer of T-depth can then be defined as

$$\begin{aligned} R^{CNOT}_{U} &\approx Cl^{CNOT}_{U}/(Q^{log}_{U} \cdot T^{d}_{SI}), \\ R^{1QC}_{U} &\approx Cl^{1QC}_{U}/(Q^{log}_{U} \cdot T^{d}_{SI}). \end{aligned} \tag{6.15}$$

A CNOT gate takes 2 surface code cycles, the single qubit gates can be evaluated in one cycle and Hadamard gates need surface code cycles equal to the code distance [Fow+12]. The required surface code cycles to implement the Clifford gates σ_{Cl} can then be determined by the ratio of the corresponding Clifford gate times the needed surface code cycles. For all considered attacks, the required surface code cycles to implement the Clifford gates σ_{Cl} are substantially lower than the surface code cycles needed for the magic state distillation σ_{dist}. Hence, most of the qubits performing the Clifford gates will be idle most of the σ_{dist} cycles. The production of the magic states is the most limiting step in the process. Therefore, only the process of providing the logical T-gates with the purified magic states has to be taken into account when determining σ_{SA}, the final number of surface code cycles.

$$\sigma_{SA} = \lambda_{SA} \cdot \sigma_{dist} \cdot T_U^d. \tag{6.16}$$

The final cost estimate is then equal to the total number of logical qubits and the number of code cycles.

$$cost_U = \left(Q_U^{log} + Q_{dist}^{log}\right) \cdot \sigma_{SA} \tag{6.17}$$

6.4 Key recovery attacks on Elephant

For the attacks presented in section 5.2 the resource estimation for the quantum attack circuit by Bonnetain in [BJ20] were used and can be found in Table 6.1 below.

Instance	Qubits	T-gates	#CNOT	#1QC	T-depth	Full depth
Elephant[160]	160	$1.76 \cdot 10^5$	$4.4 \cdot 10^5$	76677	3095	21135
Elephant[176]	176	$2.18 \cdot 10^5$	$5.5 \cdot 10^5$	95027	3605	27525
Elephant[200]	400	43581	$5.61 \cdot 10^5$	70779	343	$1.69 \cdot 10^5$

Table 6.1: Quantum circuit costs of the `Elephant` variants. 1QC denotes the single qubit Clifford operations

6.4.1 Q2 attack on Elephant v2

Setup Using the analysis of the superposition attack in subsection 5.2.2 and Bonnetain's results from [Bon20], the approximated cost of the attack can be computed. For this attack the standard version of Simon's algorithm will be used. According to Heuristic 1, it is sufficient to execute $n + 3$ iterations of Simon's algorithm to achieve a successful result. The truncated output of the `Elephant` function to 64-bits is still sufficient large to apply Simon's algorithm, as seen in the results of section 3.4 and Heuristic 4. Therefore a total of

$$\lambda_{SA} = 3 \cdot (n + 3)$$

iterations of Simon's algorithm is needed to successfully recover the hidden period s.

In each iteration of Simon's algorithm, a single call to the oracle is made. Thus, the T-gates of a single Simon iteration (SI) is approximated that of an `Elephant` call $T_{SI}^c = T_{Elephant}^c$.

The process of section 6.3 for a cost estimate are applied here and the results will be presented below. The final cost estimate can be found in Table 6.5.

Magic state distillation The output error rates p_{out} of the different `Elephant` instances are presented in Table 6.2.

Instance	T-gates	p_{out}
Elephant[160]	$8.61 \cdot 10^{7}$	$1.16 \cdot 10^{-8}$
Elephant[176]	$1.17 \cdot 10^{8}$	$8.54 \cdot 10^{-9}$
Elephant[200]	$2.65 \cdot 10^{7}$	$3.77 \cdot 10^{-8}$

Table 6.2: T-gates and the desired error rate of the different `Elephant` instances.

Using Algorithm 7 to determine the number of layers and corresponding distances of the surface codes leads to the following results for the respective instances of `Elephant` in Table 6.3 below.

Instance	p_{in}	Layer i	d_i	Q_i^{log}	Q_i^{phy}	σ_{dist}	ϕ	t_{single}
Elephant[160]	10^{-4}	1	7	16	2048	70	-	1204.9 s
	10^{-3}	1	11	16	4608	170	5	585.24 s
		2	6	240	23520			
	10^{-2}	1	23	16	18432	360	5	1239.32 s
		2	13	240	94080			
Elephant[176]	10^{-4}	1	7	16	2048	70	-	1638.92 s
	10^{-3}	1	11	16	4608	170	5	796.05 s
		2	6	240	23520			
	10^{-2}	1	24	16	20000	380	5	1779.40 s
		2	14	240	108000			
Elephant[200]	10^{-4}	1	7	16	2048	70	-	180.48 s
	10^{-3}	1	11	16	4608	170	5	371.57 s
		2	6	240	23520			
	10^{-2}	1	22	16	16928	350	5	371.57 s
		2	13	240	94080			

Table 6.3: Resource costs for the magic state distilleries.

Application Layer The number of Clifford gates is responsible for the distance of the surface code to implement the algorithm on the application layer. For all instances of `Elephant` , the most limiting step in the process is the production of the magic states. The results of the application layer can be found in Table 6.4 below.

Instance	p_{in}	$d_{Elephant}$	Q^{phy}_{SA}	T^{w}_{SI}	Φ	Q^{phy}_{total}	σ_{Cl}
Elephant[160]	10^{-4}	8	$2.59 \cdot 10^4$	56	56	$1.15 \cdot 10^5$	2
	10^{-3}	15	$8.19 \cdot 10^4$		12	$2.82 \cdot 10^5$	
	10^{-2}	173	$9.69 \cdot 10^6$		12	$1.13 \cdot 10^6$	
Elephant[176]	10^{-4}	8	$2.85 \cdot 10^4$	60	60	$1.23 \cdot 10^5$	2
	10^{-3}	15	$9.01 \cdot 10^4$		12	$2.82 \cdot 10^5$	
	10^{-2}	176	$1.10 \cdot 10^7$		12	$1.30 \cdot 10^6$	
Elephant[200]	10^{-4}	8	$6.48 \cdot 10^4$	127	127	$2.66 \cdot 10^5$	9
	10^{-3}	15	$2.05 \cdot 10^5$		26	$6.12 \cdot 10^5$	
	10^{-2}	177	$2.53 \cdot 10^7$		26	$2.45 \cdot 10^6$	

Table 6.4: Resource costs for the application layer.

Total cost estimates Taking all the results into account lead to the following Table 6.5 showing the final cost estimates.

		Elephant[160]			Elephant[176]			Elephant[200]		
		10^{-4}	10^{-3}	10^{-2}	10^{-4}	10^{-3}	10^{-2}	10^{-4}	10^{-3}	10^{-2}
Simon	T-gates		$8.61 \cdot 10^7$			$1.17 \cdot 10^8$			$2.65 \cdot 10^7$	
	T-depth		$1.51 \cdot 10^6$			$1.94 \cdot 10^6$			$2.09 \cdot 10^5$	
	Log. QB		160			176			400	
	Distance	8	15	173	8	15	176	8	15	177
	Phys. QB	$2.59 \cdot 10^4$	$8.19 \cdot 10^4$	$9.69 \cdot 10^6$	$2.85 \cdot 10^4$	$9.01 \cdot 10^4$	$1.10 \cdot 10^7$	$6.48 \cdot 10^4$	$2.05 \cdot 10^5$	$2.53 \cdot 10^7$
Distilleries	Log. QB	16	240	240	16	240	240	16	240	240
	#Distill.	56	12	12	60	12	12	127	26	26
	Distance(s)	[7]	[11, 6]	[23, 13]	[7]	[11, 6]	[24, 14]	[7]	[11]	[22, 13]
	Phys. QB	$1.15 \cdot 10^5$	$2.82 \cdot 10^5$	$1.13 \cdot 10^6$	$1.23 \cdot 10^5$	$2.82 \cdot 10^5$	$1.30 \cdot 10^6$	$2.66 \cdot 10^5$	$6.12 \cdot 10^5$	$2.45 \cdot 10^6$
Total	Log. QB	1056	3040	3040	1136	3056	3056	2432	6640	6640
	code cycles	$1.06 \cdot 10^8$	$2.57 \cdot 10^8$	$5.45 \cdot 10^8$	$1.36 \cdot 10^8$	$3.29 \cdot 10^8$	$7.36 \cdot 10^8$	$1.46 \cdot 10^7$	$3.55 \cdot 10^7$	$7.31 \cdot 10^7$
	Total time in s	21.19	51.46	108.97	27.1	65.82	147.13	2.94	7.1	14.62
	Total cost	$1.12 \cdot 10^{11}$	$7.82 \cdot 10^{11}$	$1.66 \cdot 10^{12}$	$1.54 \cdot 10^{11}$	$1.01 \cdot 10^{12}$	$2.25 \cdot 10^{12}$	$3.56 \cdot 10^{10}$	$2.36 \cdot 10^{11}$	$4.85 \cdot 10^{11}$

Table 6.5: Final cost estimates of the different `Elephant` instances using Simon's algorithm.

6.4.2 Q1 attack on Elephant v2

Setup Combining the analysis of the superposition attack in subsection 5.2.2 and Bonnetain's results from [Bon20], the approximated cost of the attack can be computed. Using Heuristic 3 results in a total of

$$\lambda_{SA} = \frac{\pi}{4 \arcsin \sqrt{2^{-k}}}$$

required iterations of Simon's algorithm with $n + k + \alpha + 4$ queries each. The values α and k are chosen as $k = n/3$ and $\alpha = 7$ to ensure a success probability of more than 99%. The cost estimate proceeds in the same way as seen in the previous sections. The final results are displayed in Table 6.9.

Magic state distillation The target output error rates p_{out} of the different `Elephant` instances are presented in Table 6.6 below.

Instance	T-gates	p_{out}
Elephant[160]	$9.91 \cdot 10^{15}$	$1.01 \cdot 10^{-16}$
Elephant[176]	$8.54 \cdot 10^{16}$	$1.17 \cdot 10^{-17}$
Elephant[200]	$3.09 \cdot 10^{17}$	$3.24 \cdot 10^{-18}$

Table 6.6: T-gates and the desired error rate of the different `Elephant` instances.

Using Algorithm 7 to determine the number of layers and corresponding distances of the surface codes leads to the following results for the respective instances of `Elephant` in Table 6.7 below.

Instance	p_{in}	Layer i	d_i	Q_i^{log}	Q_i^{phy}	σ_{dist}	ϕ	t_{single}
Elephant[160]	10^{-4}	1	13	16	6272	190	3	3977.9 years
		2	6	240	23520			
	10^{-3}	1	19	16	12800	280	6	5862.1 years
		2	9	240	48000			
	10^{-2}	1	40	16	53792	710	6	7432.2 years
		2	19	240	$1.92 \cdot 10^5$			
		3	12	3600	$1.22 \cdot 10^6$			
Elephant[176]	10^{-4}	1	13	16	6272	190	3	$3.43 \cdot 10^4$ years
		2	6	240	23520			
	10^{-3}	1	20	16	14112	290	3	$5.23 \cdot 10^4$ years
		2	9	240	48000			
	10^{-2}	1	42	16	59168	740	5	$8.01 \cdot 10^4$ years
		2	20	240	$2.12 \cdot 10^5$			
		3	12	3600	$1.22 \cdot 10^6$			
Elephant[200]	10^{-4}	1	14	16	7200	200	3	$1.30 \cdot 10^5$ years
		2	6	240	23520			
	10^{-3}	1	21	16	15488	300	3	$1.96 \cdot 10^5$ years
		2	9	240	48000			
	10^{-2}	1	43	16	61952	750	5	$2.93 \cdot 10^5$ years
		2	20	240	$2.12 \cdot 10^5$			
		3	12	3600	$1.22 \cdot 10^6$			

Table 6.7: Resource costs for the magic state distilleries.

Application Layer The number of Clifford gates is responsible for the distance of the surface code to implement the algorithm on the application layer. For all instances of `Elephant` , the most limiting step in the process is the production of the magic states. The results of the application layer can be found in Table 6.8 below.

Instance	p_{in}	$d_{Elephant}$	Q_{SA}^{phy}	T_{SI}^{w}	Φ	Q_{total}^{phy}	σ_{Cl}
Elephant[160]	10^{-4}	15	$8.19 \cdot 10^4$	56	19	$4.47 \cdot 10^5$	2
	10^{-3}	30	$3.08 \cdot 10^5$		19	$9.12 \cdot 10^5$	
	10^{-2}	339	$3.70 \cdot 10^7$		10	$1.22 \cdot 10^7$	
Elephant[176]	10^{-4}	16	$1.02 \cdot 10^5$	60	20	$4.70 \cdot 10^5$	2
	10^{-3}	31	$3.60 \cdot 10^5$		20	$9.60 \cdot 10^5$	
	10^{-2}	359	$4.56 \cdot 10^7$		12	$1.46 \cdot 10^7$	
Elephant[200]	10^{-4}	17	$2.59 \cdot 10^5$	127	43	$1.01 \cdot 10^6$	9
	10^{-3}	34	$9.80 \cdot 10^5$		43	$2.06 \cdot 10^6$	
	10^{-2}	384	$1.19 \cdot 10^8$		26	$3.16 \cdot 10^7$	

Table 6.8: Resource costs for the application layer.

Total cost estimates The following Table 6.9 displays the final cost estimates of the Q1 superposition attack on `Elephant` v2.

		Elephant[160]			Elephant[176]			Elephant[200]		
		10^{-4}	10^{-3}	10^{-2}	10^{-4}	10^{-3}	10^{-2}	10^{-4}	10^{-3}	10^{-2}
Simon	T-gates		$9.91 \cdot 10^{15}$			$8.54 \cdot 10^{16}$			$3.09 \cdot 10^{17}$	
	T-depth		$1.74 \cdot 10^{14}$			$1.41 \cdot 10^{15}$			$2.43 \cdot 10^{15}$	
	Log. QB		160			176			400	
	Distance	15	30	339	16	31	359	17	34	384
	Phys. QB	$8.19 \cdot 10^4$	$3.08 \cdot 10^5$	$3.70 \cdot 10^7$	$1.02 \cdot 10^5$	$3.60 \cdot 10^5$	$4.56 \cdot 10^7$	$2.59 \cdot 10^5$	$9.80 \cdot 10^5$	$1.19 \cdot 10^8$
Distilleries	Log. QB	240	240	3600	240	240	3600	240	240	3600
	#Distill.	19	19	10	20	20	12	43	43	26
	Distance(s)	[13, 6]	[19, 9]	[40, 19, 12]	[13, 6]	[20, 9]	[42, 20, 12]	[14, 6]	[21, 9]	[43, 20, 12]
	Phys. QB	$4.47 \cdot 10^5$	$9.12 \cdot 10^5$	$1.22 \cdot 10^7$	$4.70 \cdot 10^5$	$9.60 \cdot 10^5$	$1.46 \cdot 10^7$	$1.01 \cdot 10^6$	$2.06 \cdot 10^6$	$3.16 \cdot 10^7$
Total	Log. QB	4720	4720	36160	4976	4976	43376	10720	10720	94000
	code cycles	$3.31 \cdot 10^{16}$	$4.88 \cdot 10^{16}$	$1.24 \cdot 10^{17}$	$2.68 \cdot 10^{17}$	$4.09 \cdot 10^{17}$	$1.04 \cdot 10^{18}$	$4.86 \cdot 10^{17}$	$7.29 \cdot 10^{17}$	$1.82 \cdot 10^{18}$
	Total time in years	209.9	309.3	784.2	1699.6	2594.2	6619.6	3078.4	4617.5	$1.15 \cdot 10^4$
	Total cost	$1.56 \cdot 10^{20}$	$2.30 \cdot 10^{20}$	$4.47 \cdot 10^{21}$	$1.33 \cdot 10^{21}$	$2.04 \cdot 10^{21}$	$4.53 \cdot 10^{22}$	$5.21 \cdot 10^{21}$	$7.81 \cdot 10^{21}$	$1.71 \cdot 10^{23}$

Table 6.9: Final cost estimates of the different `Elephant` instances using offline Simon's algorithm.

6.4.3 Q2 attack on Elephant v1.1

Setup Using the results of subsection 5.2.3 and Heuristic 2 one can use $f(0) == f(s')$, where s' is the potential guess for the correct period s, as a perfect test. The attack succeeds by making approximate

$$\lambda_{SA} = \frac{\pi}{4 \arcsin \sqrt{2^{-k}}}$$

iterations with $n + \alpha/2$ queries each. The variables will be chosen as $k = (n - m)$, where $m = 96$ as discussed in subsection 5.2.3 and $\alpha = 7$ to guarantee a success probability of 99%. The final results of the attack can be seen in Table 6.13.

Magic state distillation The output error rates for the different `Elephant` instances are listed in Table 6.10 below

Instance	T-gates	p_{out}
Elephant[160]	$2.91 \cdot 10^{17}$	$3.43 \cdot 10^{-18}$
Elephant[176]	$1.01 \cdot 10^{20}$	$9.86 \cdot 10^{-21}$
Elephant[200]	$9.41 \cdot 10^{22}$	$1.06 \cdot 10^{-23}$

Table 6.10: T-gates and the desired error rate of the different `Elephant` instances.

Using the respective values for p_{out} as input for Algorithm 7, the following results for the number of layers, distances of the corresponding surface codes, number of logical and physical qubits of the magic state distillation are obtained.

Instance	p_{in}	Layer i	d_i	Q_i^{log}	Q_i^{phy}	σ_{dist}	ϕ	t_{single}
Elephant[160]	10^{-4}	1	14	16	7200	200	3	$1.23 \cdot 10^5$ years
		2	6	240	23520			
	10^{-3}	1	21	16	15488	300	3	$1.85 \cdot 10^5$ years
		2	9	240	48000			
	10^{-2}	1	43	16	61952	750	5	$2.77 \cdot 10^5$ years
		2	20	240	$2.11 \cdot 10^5$			
		3	12	3600	$1.22 \cdot 10^6$			
Elephant[176]	10^{-4}	1	15	16	8192	220	3	$4.71 \cdot 10^7$ years
		2	7	240	30720			
	10^{-3}	1	23	16	18432	390	6	$4.18 \cdot 10^7$ years
		2	10	240	58080			
		3	6	3600	$3.53 \cdot 10^5$			
	10^{-2}	1	48	16	76832	830	5	$1.07 \cdot 10^8$ years
		2	22	240	$2.54 \cdot 10^5$			
		3	13	3600	$1.41 \cdot 10^6$			
Elephant[200]	10^{-4}	1	17	16	10368	240	2	$7.16 \cdot 10^{10}$ years
		2	7	240	30720			
	10^{-3}	1	26	16	23328	430	5	$5.13 \cdot 10^{10}$ years
		2	11	240	69120			
		3	6	3600	$3.53 \cdot 10^5$			
	10^{-2}	1	54	16	96800	920	5	$1.10 \cdot 10^{11}$ years
		2	24	240	$3.00 \cdot 10^5$			
		3	14	3600	$1.62 \cdot 10^6$			

Table 6.11: Resource costs for the magic state distilleries.

Application Layer The distance of the surface code to implement the algorithm on the application layer can be obtained by considering the number of required Clifford gates. The results for the application layer can be found in Table 6.12.

Instance	p_{in}	$d_{Elephant}$	Q_{SA}^{phy}	T_{SI}^{w}	Φ	Q_{total}^{phy}	σ_{Cl}
Elephant[160]	10^{-4}	17	$1.04 \cdot 10^5$	56	19	$4.47 \cdot 10^5$	2
	10^{-3}	32	$3.48 \cdot 10^5$		19	$9.12 \cdot 10^5$	
	10^{-2}	370	$4.40 \cdot 10^7$		12	$1.46 \cdot 10^7$	
Elephant[176]	10^{-4}	19	$1.41 \cdot 10^5$	60	20	$6.14 \cdot 10^5$	2
	10^{-3}	37	$5.08 \cdot 10^5$		10	$3.53 \cdot 10^6$	
	10^{-2}	422	$6.30 \cdot 10^7$		12	$1.69 \cdot 10^7$	
Elephant[200]	10^{-4}	23	$4.61 \cdot 10^5$	127	64	$1.97 \cdot 10^6$	9
	10^{-3}	44	$1.62 \cdot 10^6$		26	$9.17 \cdot 10^6$	
	10^{-2}	498	$1.99 \cdot 10^8$		26	$4.21 \cdot 10^7$	

Table 6.12: Resource costs for the application layer.

Total cost estimates Using these results one can estimate the final cost of each attack. They can be found in Table 6.13.

		Elephant[160]			Elephant[176]			Elephant[200]		
		10^{-4}	10^{-3}	10^{-2}	10^{-4}	10^{-3}	10^{-2}	10^{-4}	10^{-3}	10^{-2}
Simon	T-gates		$2.91 \cdot 10^{17}$			$1.01 \cdot 10^{20}$			$9.41 \cdot 10^{22}$	
	T-depth		$5.12 \cdot 10^{15}$			$1.68 \cdot 10^{18}$			$7.41 \cdot 10^{20}$	
	Log. QB		160			176			400	
	Distance	17	32	370	19	37	422	23	44	498
	Phys. QB	$1.04 \cdot 10^5$	$3.48 \cdot 10^5$	$4.40 \cdot 10^7$	$1.41 \cdot 10^5$	$5.08 \cdot 10^5$	$6.30 \cdot 10^7$	$4.61 \cdot 10^5$	$1.62 \cdot 10^6$	$1.99 \cdot 10^8$
Distilleries	Log. QB	240	240	3600	240	3600	3600	240	3600	3600
	#Distill.	19	19	12	20	10	12	64	26	26
	Distance(s)	[14, 6]	[21, 9]	[43, 20, 12]	[15, 7]	[23, 10, 6]	[48, 22, 13]	[17, 7]	[26, 11, 6]	[54, 24, 14]
	Phys. QB	$4.47 \cdot 10^5$	$9.12 \cdot 10^5$	$1.46 \cdot 10^7$	$6.14 \cdot 10^5$	$3.53 \cdot 10^6$	$1.69 \cdot 10^7$	$1.97 \cdot 10^6$	$9.17 \cdot 10^6$	$4.21 \cdot 10^7$
Total	Log. QB	4720	4720	43360	4976	36176	43376	15760	94000	94000
	code cycles	$1.02 \cdot 10^{18}$	$1.54 \cdot 10^{18}$	$3.84 \cdot 10^{18}$	$3.69 \cdot 10^{20}$	$6.54 \cdot 10^{20}$	$1.39 \cdot 10^{21}$	$1.78 \cdot 10^{23}$	$3.18 \cdot 10^{23}$	$6.81 \cdot 10^{23}$
	Total time in years	6490.9	9736.3	$2.43 \cdot 10^4$	$2.34 \cdot 10^6$	$4.14 \cdot 10^6$	$8.82 \cdot 10^6$	$1.13 \cdot 10^9$	$2.02 \cdot 10^9$	$4.32 \cdot 10^9$
	Total cost	$4.83 \cdot 10^{21}$	$7.25 \cdot 10^{21}$	$1.67 \cdot 10^{23}$	$1.84 \cdot 10^{24}$	$2.37 \cdot 10^{25}$	$6.04 \cdot 10^{25}$	$2.80 \cdot 10^{27}$	$2.99 \cdot 10^{28}$	$6.41 \cdot 10^{28}$

Table 6.13: Final cost estimates of the different `Elephant` instances using the Grover-meets-Simon algorithm.

6.5 Attack on LightMAC

Setup The superposition attack on `LightMAC` by Haas in [Haa20] uses the basic version of Simon's algorithm. Therefore, using the Heuristic 1 and the results of subsection 5.3.2 the cost of the attack can be estimated. The underlying block cipher used by `LightMAC` will be AES, the basic version of Simon's algorithm will be applied, and the output is truncated hence the total number of iterations results in

$$\lambda_{SA} = 3 \cdot (2 \cdot (n + 3))$$

.
The order of the attack is exactly the same as presented in section 6.3. The used quantum circuit costs for AES were presented by Jaques et al. in [Jaq+20].

Quantum Circuit	Qubits	T-gates	#CNOT	#1QC Gates	T-depth	Full depth
AES-128 [Jaq+20]	1785	54400	291150	83116	120	2827

Table 6.14: Quantum circuit costs of AES-128.

Remark While the T-depth of the AES block cipher is given as 120, the authors of [Jaq+20] note that the anticipated T-depth would be higher. Considering n rounds, the expected T-depth would be $2n$ times the T-depth of the S-box. They assume that the lower T-depth comes from the fact that the *Q#* compiler found nontrivial parallelization between elements of the S-box and the surrounding circuit.

The used values for T-, CNOT-, 1QC-gates and T-depth will be doubled for the analysis as the block cipher is called in two instances using the function f provided in section 5.3. This leads to the following results

Instance	T-gates	p_{out}
LightMAC[AES]	$8.55 \cdot 10^7$	$1.17 \cdot 10^{-8}$

Table 6.15: T-gates and the desired error rate of the `LightMAC` instance using AES as its internal block cipher.

Magic state distillation The input for Algorithm 7 are p_{in}, $p_g = p_{in}/10$, $\epsilon = 1$ and, p_{out}. The results for the magic state process are the following.

Instance	p_{in}	Layer i	d_i	Q_i^{log}	Q_i^{phy}	σ_{dist}	ϕ	t_{single}
LightMAC[AES]	10^{-4}	1	7	16	2048	70	-	1197.24 s
	10^{-3}	1	11	16	4608	170	5	581.51 s
		2	6	240	23520			
	10^{-2}	1	23	16	18432	360	5	1231.44 s
		2	13	240	94080			

Table 6.16: Resource costs for the magic state distilleries in the `LightMAC` attack.

Comparing the overhead produced by Clifford gates per layer of T-depth to the magic state distillery one will be skipped in the following analysis, as it will not outweigh the cost of producing the magic states.

Application Layer This leads to the following results, using the same method of analysis as was used in the `Elephant` cost estimate. The width of the circuit is $Q_{LightMAC}^{log} = 1785$.

Instance	p_{in}	$d_{LightMAC}$	Q^{phy}_{SA}	T^{w}_{SI}	Φ	Q^{phy}_{total}
LightMAC	10^{-4}	8	$2.89 \cdot 10^5$	453	453	$9.28 \cdot 10^5$
	10^{-3}	15	$9.14 \cdot 10^5$		91	$2.14 \cdot 10^6$
	10^{-2}	180	$1.17 \cdot 10^8$		91	$8.56 \cdot 10^6$

Table 6.17: Resource costs for the application layer in the `LightMAC` attack.

Total cost estimates For the total costs, a single implementation of the `LightMAC` attack is first considered, and then the expected costs with respect to the parameter ν are estimated.

		LightMAC		
		10^{-4}	10^{-3}	10^{-2}
Simon	T-gates		$8.55 \cdot 10^7$	
	T-depth		$1.89 \cdot 10^5$	
	Log. QB		1785	
	Distance	8	15	180
	Phys. QB	$2.89 \cdot 10^5$	$9.14 \cdot 10^5$	$1.17 \cdot 10^8$
Distilleries	Log. QB	16	240	240
	#Distill.	453	91	91
	Distance(s)	[7]	[11, 6]	[23, 13]
	Phys. QB	$9.28 \cdot 10^5$	$2.14 \cdot 10^6$	$8.56 \cdot 10^6$
Total	Log. QB	9033	23625	23625
	code cycles	$1.32 \cdot 10^7$	$3.21 \cdot 10^7$	$6.79 \cdot 10^7$
	Total time in s	2.64	6.41	13.58
	Total cost	$1.19 \cdot 10^{11}$	$7.58 \cdot 10^{10}$	$1.60 \cdot 10^{12}$

Table 6.18: Final cost estimates of the `LightMAC` instance using AES as its internal block cipher.

6.6 Tag forgery attack on ESTATE

The superposition attack on `ESTATE` by Haas in [Haa20] uses the basic version of Simon's algorithm. Applying Heuristic 1 for the number of repetitions and the results of subsection 5.4.2 the cost of the attack can be estimated. The cost analysis will be done considering the block ciphers AES-128 and GIFT-128.

Quantum Circuit	Qubits	T-gates	#CNOT	#1QC Gates	T-depth	Full depth
GIFT-128[BCS20]	256	35840	87040	18080	160	1320
AES-128 [Jaq+20]	1785	54400	291150	83116	120	2827

Table 6.19: Quantum circuit costs of GIFT and AES.

Setup As stated in the analysis of the attack in subsection 5.4.2 at most $2n + 2$ calls of the oracles have to be made. Since the basic version of Simon's algorithm is used Heuristic 1 can be applied and the total number of iterations is therefore

$$\lambda_{SA} = 2 \cdot (n + 3) + 2$$

Two AES-128 or GIFT-128 calls have to be made in each oracle query. For this cost estimation it is assumed, that the prefix of the forged message is of length 1 and only one additional block cipher invocation has therefore to be made. This results in a total of 3 block cipher calls per oracle query. This leads to the following cost estimation. The final results can be found in Table 6.23 at the end of this section.

Magic state distillation The output error rates p_{out} are shown in Table 6.20.

Instance	T-gates	p_{out}
ESTATE[AES]	$4.28 \cdot 10^{7}$	$2.34 \cdot 10^{-8}$
ESTATE[GIFT]	$2.28 \cdot 10^{7}$	$3.55 \cdot 10^{-8}$

Table 6.20: T-gates and the desired error rate of AES and GIFT in the ESTATE attack.

Remark A stated in the section 6.5 the T-depth of the AES block cipher is given as 120, the authors of [Jaq+20] note that the anticipated T-depth would be higher. Considering n rounds, the expected T-depth would be $2n$ times the T-depth of the S-box. They assume that the lower T-depth comes from the fact that the *Q#* compiler found nontrivial parallelization between elements of the S-box and the surrounding circuit.

First the Algorithm 7 is used to determine the number of layers and corresponding distances of the surface codes. This leads to the following results shown in Table 6.21 below.

Instance	p_{in}	Layer i	d_i	Q_i^{log}	Q_i^{phy}	σ_{dist}	ϕ	t_{single}
ESTATE[AES]	10^{-4}	1	7	16	2048	70	-	598.62 s
	10^{-3}	1	11	16	4608	170	5	290.76 s
		2	6	240	23520			
	10^{-2}	1	23	16	18432	360	5	615.72 s
		2	13	240	94080			
ESTATE[GIFT]	10^{-4}	1	7	16	2048	70	-	394.38 s
	10^{-3}	1	11	16	4608	170	5	191.56 s
		2	6	240	23520			
	10^{-2}	1	22	16	16928	350	5	394.38 s
		2	13	240	94080			

Table 6.21: Resource costs for the magic state distilleries.

Application Layer The number of Clifford gates defines the distance of the surface code of the application layer. For all instances, the most limiting step in the process is the production of the magic states. The results for the attacks on ESTATE can be found in Table 6.4. The width of the circuit is $Q^{log}_{LightMAC} = 1785$ when the AES-128 block cipher is being used and $Q^{log}_{LightMAC} = 256$ when using the GIFT-128 block cipher.

Instance	p_{in}	d_{ESTATE}	Q^{phy}_{SA}	T^w_{SI}	Φ	Q^{phy}_{total}	σ_{Cl}
ESTATE[AES]	10^{-4}	8	$2.89 \cdot 10^5$	453	453	$9.28 \cdot 10^5$	3
	10^{-3}	15	$9.14 \cdot 10^5$		91	$2.14 \cdot 10^6$	
	10^{-2}	174	$1.09 \cdot 10^8$		91	$8.56 \cdot 10^6$	
ESTATE[GIFT]	10^{-4}	7	$3.28 \cdot 10^4$	224	224	$4.59 \cdot 10^5$	5
	10^{-3}	14	$1.15 \cdot 10^5$		45	$1.06 \cdot 10^6$	
	10^{-2}	163	$1.38 \cdot 10^7$		45	$4.23 \cdot 10^6$	

Table 6.22: Resource costs for the application layer.

Total cost estimates Taking all the results into account lead to the following Table 6.5, showing the final cost estimates.

		ESTATE[AES]			ESTATE[GIFT]		
		10^{-4}	10^{-3}	10^{-2}	10^{-4}	10^{-3}	10^{-2}
Simon	T-gates		$4.28 \cdot 10^7$			$2.28 \cdot 10^7$	
	T-depth		$9.43 \cdot 10^4$			$1.26 \cdot 10^5$	
	Log. QB		1785			256	
	Distance	8	15	174	7	14	163
	Phys. QB	$2.89 \cdot 10^5$	$9.14 \cdot 10^5$	$1.09 \cdot 10^8$	$3.28 \cdot 10^4$	$1.15 \cdot 10^5$	$1.38 \cdot 10^7$
Distilleries	Log. QB	16	240	240	16	240	240
	#Distill.	453	91	91	224	45	45
	Distance(s)	[7]	[11, 6]	[23, 13]	[7]	[11, 6]	[22, 13]
	Phys. QB	$9.28 \cdot 10^5$	$2.14 \cdot 10^6$	$8.56 \cdot 10^6$	$4.59 \cdot 10^5$	$1.06 \cdot 10^6$	$4.23 \cdot 10^6$
Total	Log. QB	9033	23625	23625	3840	11056	11056
	code cycles	$6.60 \cdot 10^6$	$1.60 \cdot 10^7$	$3.40 \cdot 10^7$	$8.80 \cdot 10^6$	$2.14 \cdot 10^7$	$4.40 \cdot 10^7$
	Total time in s	1.32	3.21	6.79	1.76	4.28	8.80
	Total cost	$5.96 \cdot 10^{10}$	$3.79 \cdot 10^{11}$	$8.02 \cdot 10^{11}$	$3.38 \cdot 10^{10}$	$2.36 \cdot 10^{11}$	$4.87 \cdot 10^{11}$

Table 6.23: Final cost estimates of the ESTATE attack using Simon's algorithm.

7 Conclusion

We proposed multiple attacks on lightweight cryptographic primitives using different variants of Simon's algorithm. Theses included three new quantum key recovery attacks on the `NIST` Lightweight Cryptography Standardization Process finalist `Elephant` as a result of this thesis. We provided analyses of truncated 2-to-1 functions and concluded that Simon's algorithm is still applicable. This result was applicable on already existing superposition attacks providing an exponential speed-up.

We then estimated the required resources and time of all presented quantum superposition attacks using a fault-tolerant surface code based quantum computer. These results are not optimized and are therefore no lower bound. Nevertheless, they showed that the additional resource cost overhead is immense when quantum error correcting code is considered. This shows the importance of efficient fault-tolerant quantum error correcting code when considering large scale quantum computations.

Another important conclusion of this thesis is the difference between an adversary that is able to query the primitives in superposition and one that only performs classical queries. Even when using the best known attacks targeting the same primitive will still differ in the order of 10^{10} logical-qubit-cycles.

The given cost estimates were considered for realistic and optimistic per-gate error rates from 10^{-3} to 10^{-5}, highlighting the differences between the resulting costs.

Acknowledgement

I would like to thank my supervisors Marcel Tiepelt and Astrid Ottenhues for the advice, discussions, and support. I would also like to thank my family for their support and help during this thesis.

Bibliography

[01] *Specification for the Advanced Encryption Standard (AES).* Federal Information Processing Standards Publication 197. 2001. URL: `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[AB08] Dorit Aharonov and Michael Ben-Or. "Fault-tolerant quantum computation with constant error rate". In: *SIAM Journal on Computing* (2008).

[Amy+17] Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, and John Schanck. "Estimating the Cost of Generic Quantum Pre-image Attacks on SHA-2 and SHA-3". In: *Selected Areas in Cryptography – SAC 2016.* Ed. by Roberto Avanzi and Howard Heys. Cham: Springer International Publishing, 2017, pp. 317–337. ISBN: 978-3-319-69453-5.

[Ban+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. "GIFT: a small present". In: *International Conference on Cryptographic Hardware and Embedded Systems.* Springer. 2017, pp. 321–345.

[Bas+18] Lawrence Bassham, Çağdaş Çalık, Kerry McKay, and Meltem Sönmez Turan. "Submission requirements and evaluation criteria for the lightweight cryptography standardization process". In: *US National Institute of Standards and Technology* (2018).

[BCS20] Subodh Bijwe, Amit Kumar Chauhan, and Somitra Kumar Sanadhya. "Quantum Search for Lightweight Block Ciphers: GIFT, SKINNY, SATURNIN". In: *Cryptology ePrint Archive* (2020).

[Bea02] Stephane Beauregard. "Circuit for Shor's algorithm using 2n+ 3 qubits". In: *arXiv preprint quant-ph/0205095* (2002).

[Ben80] Paul Benioff. "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines". In: *Journal of statistical physics* 22.5 (1980), pp. 563–591.

[Ber+11] Guido Bertoni, Michaël Peeters, Gilles Van Assche, and et al. *The KECCAK reference.* 2011. URL: `https://keccak.noekeon.org/Keccak-reference-3.0.pdf`.

[Ber99] Daniel J Bernstein. "How to stretch random functions: the security of protected counter sums". In: *Journal of Cryptology* 12.3 (1999), pp. 185–192.

[BJ20] Xavier Bonnetain and Samuel Jaques. *Quantum Period Finding against Symmetric Primitives in Practice.* 2020. arXiv: `2011.07022 [quant-ph]`.

[BK05a] Sergey Bravyi and Alexei Kitaev. "Universal quantum computation with ideal Clifford gates and noisy ancillas". In: *Physical Review A* 71.2 (2005), p. 022316.

[BK05b] Sergey Bravyi and Alexei Kitaev. "Universal quantum computation with ideal Clifford gates and noisy ancillas". In: *Phys. Rev. A* 71 (2 Feb. 2005), p. 022316. DOI: `10.1103/PhysRevA.71.022316`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.71.022316`.

[Bog+07] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. "PRESENT: An Ultra-Lightweight Block Cipher". In: *Cryptographic Hardware and Embedded Systems - CHES 2007*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466. ISBN: 978-3-540-74735-2.

[Bog+11] Andrey Bogdanov, Miroslav Knežević, Gregor Leander, Deniz Toz, Kerem Varıcı, and Ingrid Verbauwhede. "spongent: A Lightweight Hash Function". In: *Cryptographic Hardware and Embedded Systems – CHES 2011*. Ed. by Bart Preneel and Tsuyoshi Takagi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 312–325. ISBN: 978-3-642-23951-9.

[Bon+19] Xavier Bonnetain, Akinori Hosoyamada, Marıa Naya-Plasencia, Yu Sasaki, and André Schrottenloher. "Quantum attacks without superposition queries: the offline Simon's algorithm". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2019, pp. 552–583.

[Bon19] Xavier Bonnetain. "Hidden Structures and Quantum Cryptanalysis". Theses. Sorbonne Université, Nov. 2019. URL: `https://tel.archives-ouvertes.fr/tel-02400328`.

[Bon20] Xavier Bonnetain. *Tight Bounds for Simon's Algorithm*. Cryptology ePrint Archive, Report 2020/919. `https://ia.cr/2020/919`. 2020.

[BR05] John Black and Phillip Rogaway. "CBC MACs for arbitrary-length messages: The three-key constructions". In: *Journal of cryptology* 18.2 (2005), pp. 111–131.

[Bra+02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. "Quantum amplitude amplification and estimation". In: *Contemporary Mathematics* 305 (2002), pp. 53–74.

[Bru20] Todd A. Brun. *Quantum Error Correction*. Feb. 2020. DOI: `10.1093/acrefore/9780190871994.013.35`.

[BT21] Robin M. Berger and Marcel Tiepelt. *On Forging SPHINCS+-Haraka Signatures on a Fault-tolerant Quantum Computer*. Cryptology ePrint Archive, Report 2021/1484. `https://ia.cr/2021/1484`. 2021.

[BZ13] Dan Boneh and Mark Zhandry. "Secure Signatures and Chosen Ciphertext Security in a Quantum Computing World". In: *Advances in Cryptology – CRYPTO 2013*. Ed. by Ran Canetti and Juan A. Garay. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 361–379. ISBN: 978-3-642-40084-1.

[Cha+20] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas-López, Mridul Nandi, and Yu Sasaki. “ESTATE: A lightweight and low energy authenticated encryption mode”. In: *IACR Transactions on Symmetric Cryptology* (2020), pp. 350–389.

[DA07] David P DiVincenzo and Panos Aliferis. “Effective fault-tolerant quantum computation with slow measurements”. In: *Physical review letters* 98.2 (2007), p. 020501.

[Dae91] Joan Daemen. “Limitations of the Even-Mansour construction”. In: *International Conference on the Theory and Application of Cryptology*. Springer. 1991, pp. 495–498.

[Dam+11] Ivan Damgaard, Jakob Funder, Jesper Buus Nielsen, and Louis Salvail. *Superposition Attacks on Cryptographic Protocols*. 2011. arXiv: `1108.6313 [quant-ph]`.

[DM19] Christoph Dobraunig and Bart Mennink. “Elephant v1.” In: (2019). URL: `https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/elephant-spec-round2.pdf`.

[DMN13] Simon J Devitt, William J Munro, and Kae Nemoto. “Quantum error correction for beginners”. In: *Reports on Progress in Physics* 76.7 (June 2013), p. 076001. ISSN: 1361-6633. DOI: `10.1088/0034-4885/76/7/076001`. URL: `http://dx.doi.org/10.1088/0034-4885/76/7/076001`.

[Dwo+16] Morris Dworkin et al. “Recommendation for block cipher modes of operation: methods for format-preserving encryption”. In: *NIST Special Publication* 800 (2016), 38G.

[EM97] Shimon Even and Yishay Mansour. “A construction of a cipher from a single pseudorandom permutation”. In: *Journal of cryptology* 10.3 (1997), pp. 151–161.

[FDJ13] Austin G Fowler, Simon J Devitt, and Cody Jones. “Surface code implementation of block code state distillation”. In: *Scientific reports* 3.1 (2013), pp. 1–6.

[Fow+12] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. “Surface codes: Towards practical large-scale quantum computation”. In: *Physical Review A* 86.3 (2012), p. 032324.

[FSG09] Austin G Fowler, Ashley M Stephens, and Peter Groszkowski. “High-threshold universal quantum computation on the surface code”. In: *Physical Review A* 80.5 (2009), p. 052312.

[Gag17] Tommaso Gagliardoni. “Quantum security of cryptographic primitives”. In: *arXiv preprint arXiv:1705.02417* (2017).

[GE21] Craig Gidney and Martin Ekerå. “How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits”. In: *Quantum* 5 (Apr. 2021), p. 433. ISSN: 2521-327X. DOI: `10.22331/q-2021-04-15-433`. URL: `https://doi.org/10.22331/q-2021-04-15-433`.

[Gib19] Elizabeth Gibney. “The quantum gold rush”. In: *Nature* 574.7776 (2019), pp. 22–24.

[Got06] Daniel Gottesman. “Quantum error correction and fault-tolerance”. In: *Quantum Information Processing: From Theory to Experiment* 199 (2006), p. 159.

[Got98a] Daniel Gottesman. “The Heisenberg representation of quantum computers”. In: *arXiv preprint quant-ph/9807006* (1998).

[Got98b] Daniel Gottesman. “Theory of fault-tolerant quantum computation”. In: *Phys. Rev. A* 57 (1 Jan. 1998), pp. 127–137. doi: 10.1103/PhysRevA.57.127. url: https://link.aps.org/doi/10.1103/PhysRevA.57.127.

[Gra+16a] Robert Granger, Philipp Jovanovic, Bart Mennink, and Samuel Neves. “Improved Masking for Tweakable Blockciphers with Applications to Authenticated Encryption”. In: *Advances in Cryptology – EUROCRYPT 2016*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 263–293. isbn: 978-3-662-49890-3.

[Gra+16b] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. “Applying Grover’s Algorithm to AES: Quantum Resource Estimates”. In: *Post-Quantum Cryptography*. Ed. by Tsuyoshi Takagi. Cham: Springer International Publishing, 2016, pp. 29–43. isbn: 978-3-319-29360-8.

[Gro96] Lov K Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 212–219.

[Haa20] Jonas Haas. “Superposition Attacks on Lightweight Message Authentication”. masters thesis. Karlsruher Institute of Technologies, 2020.

[Isa+08] Nemanja Isailovic, Mark Whitney, Yatish Patel, and John Kubiatowicz. “Running a quantum circuit at the speed of data”. In: *ACM SIGARCH Computer Architecture News* 36.3 (2008), pp. 177–188.

[Jan+21] Kyungbae Jang, Gyeongju Song, Hyunjun Kim, Hyeokdong Kwon, Hyunji Kim, and Hwajeong Seo. “Efficient Implementation of PRESENT and GIFT on Quantum Computers”. In: *Applied Sciences* 11.11 (2021). issn: 2076-3417. doi: 10.3390/app11114776. url: https://www.mdpi.com/2076-3417/11/11/4776.

[Jaq+20] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. “Implementing Grover Oracles for Quantum Key Search on AES and LowMC”. In: *Advances in Cryptology – EUROCRYPT 2020*. Ed. by Anne Canteaut and Yuval Ishai. Cham: Springer International Publishing, 2020, pp. 280–310. isbn: 978-3-030-45724-2.

[Jon+12] N. Cody Jones, Rodney Van Meter, Austin G. Fowler, Peter L. McMahon, Jungsang Kim, Thaddeus D. Ladd, and Yoshihisa Yamamoto. “Layered Architecture for Quantum Computing”. In: *Phys. Rev. X* 2 (3 July 2012), p. 031007. doi: 10.1103/PhysRevX.2.031007. url: https://link.aps.org/doi/10.1103/PhysRevX.2.031007.

[Kap+16] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. "Breaking Symmetric Cryptosystems Using Quantum Period Finding". In: *Advances in Cryptology – CRYPTO 2016*. Ed. by Matthew Robshaw and Jonathan Katz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 207–237. ISBN: 978-3-662-53008-5.

[KL15] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. 2. ed. Chapman & Hall/CRC cryptography and network security. Includes bibliographical references and index. Boca Raton, FL [u.a.]: CRC Press, 2015. ISBN: 9781466570269. URL: `http://swbplus.bsz-bw.de/bsz416906737inh.htm%20;%20https://zbmath.org/?q=an:1323.94001`.

[KM10] Hidenori Kuwakado and Masakatu Morii. "Quantum distinguisher between the 3-round Feistel cipher and the random permutation". In: *2010 IEEE International Symposium on Information Theory*. 2010, pp. 2682–2685. DOI: `10.1109/ISIT.2010.5513654`.

[KM12] Hidenori Kuwakado and Masakatu Morii. "Security on the quantum-type Even-Mansour cipher". In: *2012 International Symposium on Information Theory and its Applications*. 2012, pp. 312–316.

[Kni05] Emanuel Knill. "Quantum computing with realistically noisy devices". In: *Nature* 434.7029 (2005), pp. 39–44.

[KR96] Joe Kilian and Phillip Rogaway. "How to protect DES against exhaustive key search". In: *Annual International Cryptology Conference*. Springer. 1996, pp. 252–267.

[LM17] Gregor Leander and Alexander May. "Grover Meets Simon – Quantumly Attacking the FX-construction". In: *Advances in Cryptology – ASIACRYPT 2017*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Cham: Springer International Publishing, 2017, pp. 161–178. ISBN: 978-3-319-70697-9.

[Luy+16] Atul Luykx, Bart Preneel, Elmar Tischhauser, and Kan Yasuda. "A MAC mode for lightweight block ciphers". In: *International Conference on Fast Software Encryption*. Springer. 2016, pp. 43–59.

[Men21] Bart Mennink. "Elephant v2". In: (2021). URL: `https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/elephant-spec-final.pdf`.

[NC01] Michael A Nielsen and Isaac L Chuang. "Quantum computation and quantum information". In: *Phys. Today* 54.2 (2001), p. 60.

[PS11] Sergey Panasenko and Sergey Smagin. "Lightweight cryptography: Underlying principles and approaches". In: *International Journal of Computer Theory and Engineering* 3.4 (2011), p. 516.

[RHG06] R. Raussendorf, J. Harrington, and K. Goyal. "A fault-tolerant one-way quantum computer". In: *Annals of Physics* 321.9 (July 2006), pp. 2242–2270. ISSN: 0003-4916. DOI: `10.1016/j.aop.2006.01.012`. URL: `http://dx.doi.org/10.1016/j.aop.2006.01.012`.

[RHG07] Robert Raussendorf, Jim Harrington, and Kovid Goyal. “Topological fault-tolerance in cluster state quantum computation”. In: *New Journal of Physics* 9.6 (2007), p. 199.

[Shi+21] Tairong Shi, Wenling Wu, Bin Hu, Jie Guan, and Sengpeng Wang. “Breaking LWC candidates: sESTATE and Elephant in quantum setting”. In: *Designs, Codes and Cryptography* (2021), pp. 1–28.

[Sho94] Peter W Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[Sho95] Peter W. Shor. “Scheme for reducing decoherence in quantum computer memory”. In: *Phys. Rev. A* 52 (4 Oct. 1995), R2493–R2496. DOI: `10.1103/PhysRevA.52.R2493`. URL: `https://link.aps.org/doi/10.1103/PhysRevA.52.R2493`.

[Sim97] Daniel R. Simon. “On the Power of Quantum Computation”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1474–1483. DOI: `10.1137/S0097539796298637`. eprint: `https://doi.org/10.1137/S0097539796298637`. URL: `https://doi.org/10.1137/S0097539796298637`.

[Sön+21] Meltem Sönmez Turan, Kerry McKay, Donghoon Chang, Çağdaş Çalık, Lawrence Bassham, Jinkeon Kang, and John Kelsey. *Status report on the second round of the NIST lightweight cryptography standardization process*. Tech. rep. National Institute of Standards and Technology, 2021.

[SS16] Thomas Santoli and Christian Schaffner. “Using Simon’s algorithm to attack symmetric-key cryptographic primitives”. In: *arXiv preprint arXiv:1603.07856* (2016).

[SS17] Thomas Santoli and Christian Schaffner. *Using Simon’s Algorithm to Attack Symmetric-Key Cryptographic Primitives*. 2017. arXiv: `1603.07856 [quant-ph]`.

[ST16] National Institute of Standards and Technology. *Post-Quantum Cryptography Standardization*. `https://competitions.cr.yp.to/caesar.html`. July 2016.

[ST20a] National Institute of Standards and Technology. *Lightweight Cryptography Standardization*. `https://csrc.nist.gov/Projects/lightweight-cryptography/finalists`. July 2020.

[ST20b] National Institute of Standards and Technology. *Post-Quantum Cryptography Standardization*. `https://csrc.nist.gov/Projects/post-quantum-cryptography/Post-Quantum-Cryptography-Standardization`. July 2020.

[Ter15] Barbara M. Terhal. “Quantum error correction for quantum memories”. In: *Reviews of Modern Physics* 87.2 (Apr. 2015), pp. 307–346. ISSN: 1539-0756. DOI: `10.1103/revmodphys.87.307`. URL: `http://dx.doi.org/10.1103/RevModPhys.87.307`.

[The16] Institut für Theoretische Informatik Arbeitsgruppe für Kryptographie und Sicherheit. *Skript zur Stammvorlesung Sicherheit*. 2016.

[Wan+09] David S Wang, Austin G Fowler, Ashley M Stephens, and Lloyd Christopher L Hollenberg. “Threshold error rates for the toric and surface codes”. In: *arXiv preprint arXiv:0905.0531* (2009).

[Wol19] Ronald de Wolf. “Quantum Computing: Lecture Notes, 2011”. In: *Lecture notes for the 2011 course on Quantum Computing at the University of Amsterdam, available online at http://homepages. cwi. nl/˜ rdewolf/qcnotes. pdf* (2019).

[WZ82] William K Wootters and Wojciech H Zurek. “A single quantum cannot be cloned”. In: *Nature* 299.5886 (1982), pp. 802–803.

[Zal06] Christof Zalka. “Shor’s algorithm with fewer (pure) qubits”. In: *arXiv preprint quant-ph/0601097* (2006).

[Zou+20] Jian Zou, Zihao Wei, Siwei Sun, Ximeng Liu, and Wenling Wu. “Quantum Circuit Implementations of AES with Fewer Qubits”. In: *Advances in Cryptology – ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Cham: Springer International Publishing, 2020, pp. 697–726. ISBN: 978-3-030-64834-3.